T.C.
KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
DOKTORA TEZİ

ELEKTRONİK BELGE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ VE KÜÇÜK BOYUTLU DOSYALARDAN OLUŞAN BÜYÜK VERİNİN DEPOLANMASI İÇİN DAĞITIK DOSYA SİSTEMİ TASARIMI

MAHMUT ÜNVER

TEMMUZ 2018

Bilgisayar Mühendisliği Anabilim Dalında Mahmut ÜNVER tarafından hazırlanan ELEKTRONİK BELGE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ VE KÜÇÜK BOYUTLU DOSYALARDAN OLUŞAN BÜYÜK VERİNİN DEPOLANMASI İÇİN DAĞITIK DOSYA SİSTEMİ TASARIMI adlı Doktora Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof.Dr. Hasan ERBAY
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Doktora Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Dr.Öğr.Üyesi Atilla ERGÜZEN
Danışman

Jüri Üyeleri

Başkan : (Prof.Dr. Mehmet Reşit TOLUN) ____
Üye (Danışman) : (Dr.Öğr.Üyesi Atilla ERGÜZEN) ____
Üye : (Dr.Öğr.Üyesi Abdullah Erdal TÜMER)_
Üye : (Dr.Öğr.Üyesi Halil Murat ÜNVER)_
Üye : (Dr.Öğr.Üyesi Murat LÜY) ___

24/07/2018

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Doktora derecesini onaylamıştır.

Prof.Dr. Mustafa YİĞİTOĞLU
Fen Bilimleri Enstitüsü Müdürü

ÖZET

ELEKTRONİK BELGE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ VE KÜÇÜK BOYUTLU DOSYALARDAN OLUŞAN BÜYÜK VERİNİN DEPOLANMASI İÇİN DAĞITIK DOSYA SİSTEMİ TASARIMI

ÜNVER, Mahmut
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Doktora tezi
Danışman: Dr. Öğr. Üyesi Atilla ERGÜZEN
Temmuz 2018, 112 sayfa

Kurum ve kuruluşların işleyişleri ile ilgili işlemler sonucu ortaya big data (büyük veri) diye adlandırılan, klasik veri yönetme ve depolama yöntemleri ile depolanamayan ve sürdürülemeyen bir büyüklükte veri ortaya çıkmaktadır. Bu büyük veri, kuruluşların işlem hacminin büyük olması, ortaya çıkan elektronik belgelerin kalitesinin iyi olması ve bilişim teknolojisi aygıtlarının eskiye göre maliyetinin daha düşük olması nedeniyle kullanılabilirlik oranının artmasından kaynaklanmaktadır. Ortaya çıkan büyük veri, klasik depolama yöntemleri yerine performansı ve küçük-orta ölçekli birimler için maliyeti daha düşük olan farklı yöntemlerle yönetilmektedir. Bu çalışmada, orta ölçekli bir kuruluş için dağıtık, hybrid bir dosya yönetim sistemi tasarlanmıştır. İlk önce Elektronik Belge Yönetim Sistemi (EBYS) geliştirilmiş, daha sonra, bu yazılımın ortaya çıkarttığı veriyi yönetecek olan dağıtık dosya sistemi tasarlanmıştır. Yeni tasarlanan dosya sistemi yaygın kullanılan diğer dosya sistemleri ile karşılaştırılmıştır. Tasarlanan sistemi, diğer sistemlerden ayıran en büyük özellik sunucusuz çalışıyor olmasıdır. Diğer sistemlerde bulunan ve sunucu gibi çalışan name node, head node veya server node tasarlanan sistemde bulunmamaktadır. Bu sayede sisteme süreklilik kazandırılmakta, bir data node birden fazla istemci programa hizmet verebilmekte, en uygun yük dengeleme yapılabilmektedir. Sistemde dağıtık olarak kullanılan veri bloğu bunch

olarak adlandırılmıştır. Bir bunch 10 MB büyüklüğündedir. Tasarlanan sistemde sadece replica node ve data node kullanılmış, data node sayısı dört olarak belirlenmiştir. Bu değer, yatay ölçeklenebilirlik sayesinde sınırsız olarak arttırılabilmektedir. Bir data bloğu üç replica node 'ta tutulmakta, bu sayede veri güvenliği sağlanmaktadır. İlave olarak, sistemin güvenliği hem IP'ler kontrol edilerek hem de data node'lara gönderilen veriler şifrelenerek sağlanmaktadır. Tasarlanan dağıtılmış dosya sisteminin performansı diğer dosyalama sistemleri ile karşılaştırıldığında, İşletim Sisteminin dosya sistemine göre %74 daha iyi performans göstermiştir. Aynı şekilde, ilişkisel veri tabanı programına göre %80 ve NoSQL sisteme göre %97 daha iyi performans değerleri ortaya çıkmıştır. Tasarlanan sistem sayesinde, kurumların maliyeti, iş yükü ve teknoloji ihtiyaçları en aza indirilecektir. Tasarlanan sistem iyi bir performansa sahip, ölçeklenebilir, bazı node'larda hata oluşması durumunda bile çalışmasını sürdürebilir, name node olmayan, güvenli ve düşük maliyetli bir sistem olmuştur.

Anahtar Kelimeler: Big Data, Dağıtık Dosya Sistemi, Elektronik Belge Yönetim Sistemi, NoSQL, Hadoop, İlişkisel Veri Tabanı Yönetimi, İşletim Sistemi.

ABSTRACT

DISTRIBUTED FILE SYSTEM DESIGN FOR MANAGING AND STORING BIG DATA PRODUCED WITH ELECTRONIC DOCUMENT MANAGEMENT SYSTEM DESIGN

ÜNVER, Mahmut
Kırıkkale University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering, Ph. D. Thesis
Supervisor: Asst. Prof. Dr. Atilla ERGÜZEN
July 2018, 112 pages

As a result of the operations related to the operation of institutions and organizations, large data is generated which can not be stored and stored with conventional data management and storage methods called big data. The large transaction volume of these organizations is due to the fact that the quality of the resulting electronic data is good and therefore the data volume is large and the usability rate is increased due to the lower cost of information technology devices compared to the old ones. The resulting big data is now managed by different methods, with lower performance for small and medium-sized units, instead of classical storage methods. We designed a distributed, hybrid file management system for a medium-sized organization. First we developed Electronic Record Management System (ERMS). We then designed the distributed file system. We have compared the newly designed file system with other commonly used file systems. The main difference between the system and other systems is that it works without a server. There aren't a name node, a head node, or a server node running as a server as that are other systems. In the system, a data node can serve more than one client program. In this way, the optimum load balancing can be done. The data block used in the system is called bunch, and a bunch of 10 MB in size. Only replica nodes and data nodes are used in the system.

The number of data nodes is 4. With this horizontal scalability, unlimited values can be increased. A data block is kept at 3 replica nodes. In this way, data security is provided. In addition, the security of the system is ensured both by controlling the IPs and by encrypting the data sent to the data nodes. The designed distributed file system performance was 74% better than the operating system's file system when compared to other filing systems. Likewise, performance scores of 80% and 97% better than those of the relational database program and the NoSQL system, respectively. Thanks to the system, the cost, workload and technology needs of the institutions will be minimized. The system has been a safe, low-cost system, a good performance, scalable, non-name-node, which can continue to function even in the event of some node failure.

Key Words: Big Data, Distributed File System, Electronic Record Management System, NoSQL, Hadoop, Relational Database Management, Operating System.

TEŞEKKÜR

Tezimin hazırlanması esnasında hiçbir yardımı esirgemeyen, araştırmacılara büyük destek olan ve bilimsel konularda daima yardımını gördüğüm tez yöneticisi danışmanım, Sayın Dr.Öğr.Üyesi Atilla ERGÜZEN'e ve tez çalışması aşamasında ve her konuda destek olan Sayın Dr.Öğr.Üyesi Erdal ERDAL'a teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

ÇİZELGELER DİZİNİ

ÇİZELGE Sayfa

KISALTMALAR DİZİNİ

BT	Bilişim Teknolojileri
BTG	Bilgisayarlı Tomografi
DCCP	The Datagram Congestion Control Protocol
DSA	Digital Signature Algorithm
EB	Exabyte (10^{18} Byte)
EBYS	Elektronik Belge Yönetim Sistemi
GB	Gigabyte (10^{9} Byte)
GFS	Google File System
IoT	Internet of Things
IP	İnternet Protokol
ISO	International Organization for Standardization
İVTYS	İlişkisel Veri Tabanı Yönetim Sistemi
KB	Kilobyte (10^{3} Byte)
KÜADDS	Kırıkkale Üniversitesi Akıllı Dağıtık Dosya Sistemi
MB	Megabyte (10^{6} Byte)
MRI	Magnetic Rezonans Imaging
MTRA	Multi-Tier Resource Allocation
NFS	Network File System
NoSQL	Not Only Structured Query Language

PB	Petabyte (10^{15} Byte)
PGP	Pretty Good Privacy (PGP)
RADOS	Reliable Autonomic Distributed Object Store
RAM	Read Only Memory
ROI	Region of Interest
RPC	Remote Procedure Call
RSA	Rivest-Shamir-Adleman
SaaS	Software as a Service
SCTP	Stream Control Transport Protocol
TB	Terabyte (10^{12} Byte)
TCP	Transmission Control Protocol
TSE	Türk Standartları Enstitüsü
UDP	User Datagram Protocol
XML	Extensible Markup Language
ZB	Zetabyte (10^{21} Byte)

1. GİRİŞ

Son yıllarda hem internetin kullanımın artması hem de bilgi teknolojisindeki gelişmeler dünya da daha çok verinin üretilmesi sonucunu ortaya çıkartmıştır. Özellikle internet kullanımı, tüm alanlarda üretilen veri miktarlarını katlanarak artırmıştır. İnternet kullanıcılarının sayısı 1995 yılında 16 milyon iken bu sayı 2000 yılında 304 milyona, 2005 yılında 888 milyona, 2010 yılında 1,99 milyara, 2015 yılında 3,27 milyara ve 2017 yılında 3,885 milyara ulaşmıştır [1]. Aynı şekilde web sitesi sayısı da yıllar itibarıyla hızla artmıştır. 1995 yılında 23.500 adet web sitesi yayında iken bu sayı, 2000 yılında 17 milyona, 2005 yılında 64 milyona, 2010 yılında 209 milyona, 2015 yılında 863 milyona ve 2017 yılında 1,7 milyara ulaşmıştır. Aynı şekilde, internet kullanıcısı başına düşen web sitesi sayısı, 1995 yılında 1.908 iken, bu sayı 2000 yılında 24'e, 2010 yılında 9,9'a ve 2015 yılında 3,7'ye düşmüştür [2]. Facebook kullanıcı sayısı, 2004'te bir milyon iken bu sayı, 2012'de bir milyarı, 2017 yılında da iki milyarı aşmıştır [3].

Özellikle internet kullanımının artması tüm alanlarda üretilen veri miktarının da katlanarak artırmasına neden olmuştur. Bunda en önemli faktör, verinin kullanıcılar arasında daha kolay iletilebiliyor olmasıdır. Aynı şekilde, kullanılan bilişim teknolojilerinin maliyetlerinin düşmesi, insanların bilgiye erişimini arttırmış ve bunun sonucunda da veri hacmini arttırmıştır. IBM'in 2013 yılında yayınladığı makalede dünyada her gün, 2,5 exabyte (EB) veri üretildiği ve bu verinin de %90'ının son iki yılda üretildiği belirtilmiştir [4]. 1992 yılında bir günde 100 gigabyte (GB) veri üretilirken, 1997 yılında aynı veri büyüklüğü bir saatte, 2002 yılında bir saniyede üretilmiştir. 2013 yılında ise bir saniyede 28 terabyte (TB) veri üretilmiştir [5]. 1986-2011 yılları arasında dünya da toplam üretilen veri sayısı 295 EB'tır. 2013'te bu sayı 4,4 ZB'a, 2016 yılında 16,1 ZB'a ulaşmıştır. 2020 yılında, 44 ZB'a ve 2025 yılında da 163 ZB'a ulaşacağı tahmin edilmektedir [6].

Üretilen veriler havacılık, meteoroloji, Internet of Things (IoT) uygulamaları, sağlık, enerji, endüstri sektörleri, eğitim gibi birçok farklı alanda bulunmaktadır. Aynı

şekilde, sosyal medya aracılığıyla üretilen veriler de çok büyük hacimlere ulaşmıştır. En yaygın sosyal medya platformları çok sayıda kullanıcıya sahiptir. Ağustos-2017'de, Facebook yaklaşık 2,0 milyar, YouTube 1,5 milyar, WhatsApp 1,2 milyar, Instagram 700 milyon, Twitter 328 milyon ve LinkedIn 106 milyon aktif kullanıcıya sahiptir [7]. 2014 yılında Facebook.com günde 600 TB veri depolamakta, bu veriyi 300 PB büyüklüğündeki sunucularda saklamaktadır. Aynı zamanda Google aynı yılda günde 100 petabayt (PB) veriyi 3 milyon dağıtık sunucuda işlediğini açıklamıştır [8, 9].

Veri üretimi, sağlık sektöründe de dikkat çekici bir oranda artmıştır. Dijital medikal görüntüleme birimlerinin yaygın kullanımı bu veri üretimini tetiklemiştir. Ayrıca, sağlık sektöründe üretilen verilerin geleneksel veri yönetimi araçları ve donanımı ile kolayca yönetilemeyeceği bir noktaya ulaşılmıştır. Sağlık hizmetleri, hastaların kayıtlarını tutarak, doktorların tanı koymalarına yardımcı olacak dijital formatta tıbbi görüntüleme çıktıları oluşturarak ve farklı taramaların sonuçlarını saklayarak büyük bir veri hacmini ortaya çıkartmıştır. Farklı veri kaynakları, hasta bilgisi, laboratuvar sonuçları, X-ışını cihazları, bilgisayarlı tomografi (BT) ve manyetik rezonans görüntüleme (MRI) gibi çeşitli yapılandırılmış ve yapılandırılmamış formatlarda veri üretmektedir. Dünya nüfusu ve ortalama insan ömrü sürekli olarak artmaktadır, bu da hizmet edilecek hasta sayısında üssel bir artış anlamına gelmektedir. Hasta sayısı arttıkça toplanan veri miktarı da önemli ölçüde artmaktadır. Ayrıca, dijital sağlık cihazlarının kalitesinin artması, yüksek yoğunluklu grafik çıktıları veri büyüklüğünü hızla artırmaktadır. 2011 yılında Amerika Birleşik Devletleri'ndeki sağlık sektörünün veri miktarı 150 EB iken, değerin 2013 yılında 153 EB'a ulaştığı görülmüş, 2020 yılında da 2,3 ZB'a ulaşacağı tahmin edilmektedir. Örneğin, Elektronik Tıbbi Kayıtlarda, 2001'den 2005'e kadar dört yılda %31 artış ve 2005'ten 2008'e 3 yılda %50'den fazla artış olmuştur [10, 11]. 1985 ile 1989 arasında neuroimaging operasyon işlemlerinin veri büyüklüğü yılda 200 GB'a ulaşırken, 2010-2014 yılları arasında 5 PB'a yükselmesi sağlık sektöründe veri artışının bir göstergesi olarak ortaya çıkmıştır [12].

Ayrıca, sağlıkta big datayı oluşturan etkenlerden birisi de biyolojik verilerdir. Gen dizilmesi ile elde edilen veri kütleleri, farklı uygulamaların ihtiyaçlarına göre özel analizlerden geçirilerek klinik gen teşhisi ile birleştirilip hastalıkların erken teşhisi ve kişiselleştirilmiş tedavisi için önemli bilgiler sağlamaktadır. İnsan geninin bir dizilimi 100-600 GB ham veri üretebilmektedir. Shenzhen'deki Çin Ulusal Genbank'ta, 1,15 milyon insan, 150.000 hayvan, bitki ve mikroorganizma örneği dahil olmak üzere 1,3 milyon örnek bulunmaktadır. 2013 yılı sonunda 10 milyon izlenebilir biyolojik örnek depolanacak ve 2015 yılı sonunda bu rakam 30 milyona ulaşacak şekilde planlanmıştır [13]. 2016 yılında resmi açılışı yapılan GenBank, biyolojik bilgileri depolamak ve yorumlamak için yüksek performanslı bir veri yönetim sistemi kurmaya odaklanmıştır. İlk aşamasında 20 PB erişim kapasitesine ulaşmış ve ikinci aşamada 500 PB'a ulaşmıştır. Bir arama motoru, veri iletimi ve bulut bilişim desteği ile 10'dan fazla veri tabanına ev sahipliği yapmaktadır [14].

Eğitim alanında teknolojinin kullanımının artmasıyla üretilen veri büyüklüğü de artmıştır. Özellikle uzaktan eğitimde kullanılan elektronik ders içerikleri, canlı ders videoları, ödev dosyaları, mesajlaşma bilgileri, destekleyici dokümanlar, sunum dosyaları ve log kayıtları verinin büyük boyutlara ulaşmasına neden olmuştur. Kırıkkale Üniversitesi, Uzaktan Eğitim Merkezi kapsamında 2009 yılından beri uygulanmakta olan uzaktan eğitim siteminde, 2014-2017 yılları arasında, silinen dosyalar hariç ödevler 5 GB, canlı ders büyüklüğü 1,1 TB, ders içerikleri dosyaları 15 GB ve destekleyici diğer dokümanlar 37 GB büyüklüğe ulaşmıştır.

E-Öğrenim uzmanlarının, big data ile çalışmalarının çeşitli avantajları vardır. E-Öğrenimin geleceğini belirleme etkisine sahiptir, e-Öğrenim sürecini analiz etme ve değerlendirme yönteminde gelişmeler yapılmasını sağlar. Bunlar, büyük verilerle ilgili en önemli avantajlardan sadece birkaçıdır:

E-Öğrenim uzmanlarının, öğrenenlerin bilgiyi nasıl sindirdiğini ve hangi öğrenme gereksinimlerinin onlara faydalı olmadığını anlamalarını sağlar. Örneğin, büyük veriler, e-Öğrenim uzmanlarının, gerçekliğe dayalı bir senaryonun, metin tabanlı bir problem çözme etkinliğinden daha etkili olup olmadığını belirlemesine izin verir.

Big data analizi, E-Öğrenim uzmanlarının, e-Öğrenim kursu veya modül içinde hassas ve kişiye uygun düzenleme yapılması gerekebilecek alanları tespit etmelerini sağlar. Örneğin, birden fazla öğrenci belirli bir modülü bitirmek için aşırı zaman harcıyorsa, bu muhtemelen modülün öğrenenler için daha kolay yönetilebilmesi için geliştirilmesinin gerektiği anlamına gelmektedir.

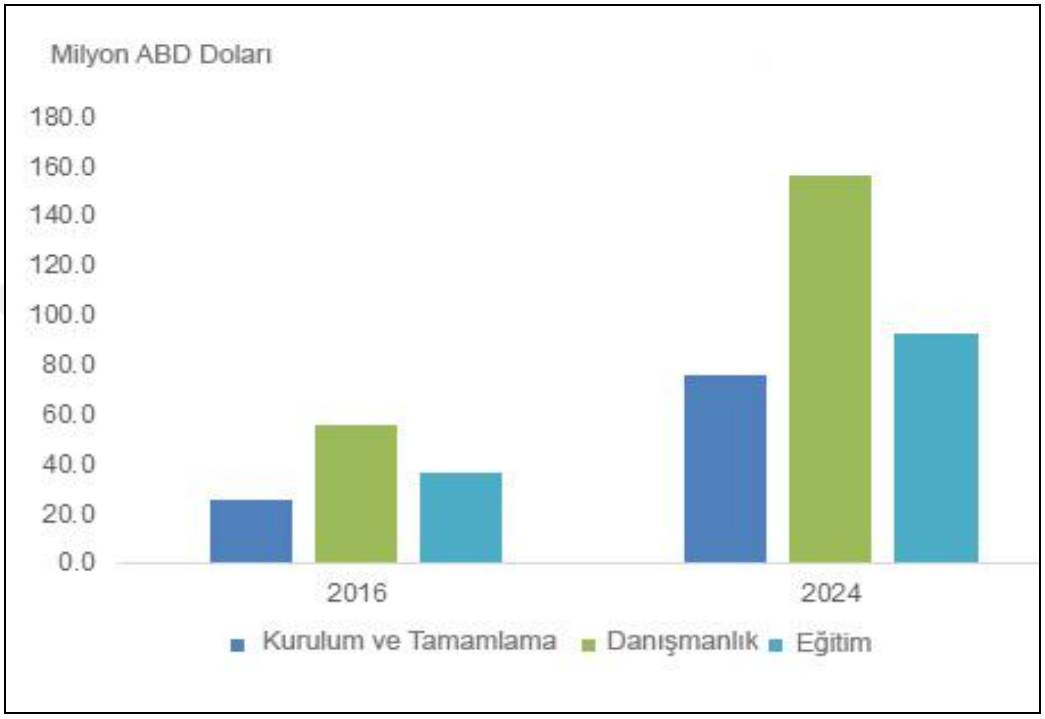
En çok hangi e-Öğrenim modüllerinin ziyaret edildiği, e-Öğrenme modüllerinin veya bağlantılarının diğer öğrenicilerle ne kadar paylaşıldığı gibi sosyal öğrenme durumlarının analizi yapılmaktadır. Örneğin, Facebook üzerinden en çok hangi bağlantının paylaşıldığı belirlenebilmektedir.

Veriler, değerlendirmeler almak için uzun süre beklemek yerine, hemen alınabilmektedir. Bu, e-Öğrenim uzmanlarının, e-Öğrenim stratejilerini doğru bir şekilde ayarlayarak değişiklikleri uygulamaya ya da verileri kullanmaya başlayabilecekleri anlamına gelir.

Big data analizi sayesinde, modellere dayalı olarak, e-Öğrenim uzmanları, öğrenenlerin mücadele edebileceği veya iyileştirilebileceği yerleri tahmin etme gücüne sahiptir. Bu şekilde, e-Öğrenim kurslarını geliştirebilirler ve böylece öğrenciler mümkün olan en iyi sonuçları elde etme şansına kavuşmuş olmaktadır.

"Veri", 2012 yılında yapılan Dünya Ekonomik Forumu'nda para ve altın gibi değeri olan bir varlık olarak tanımlanmıştır. 2011 yılında, Amerika Birleşik Devletleri'nde, 15 endüstriyel sektörde üretilen veri miktarının, en büyük kütüphanelerden biri kabul edilen Birleşik Devletler Kongre Kütüphanesi'nin sakladığı 235 TB'lık veriden daha büyük bir veri büyüklüğüne ulaştığı tespit edilmiştir. Yine aynı yılda, Wall-Mart mağazalarının sakladığı toplam verinin büyüklüğünün 2,5 PB'a ulaştığı belirtilmiştir. Şirket mağazaları saatte bir milyon müşteri verisini veri tabanlarında saklamaktadır. [15].

Elektronik Belge Yönetim Sistemi Piyasa büyüklüğü 2016 yılında 3,09 milyar Amerika Birleşik Devletleri doları iken, 2024 yılına kadar 6 milyar Amerika Birleşik Devletleri dolarını aşacağı tahmin edilmektedir. Birleşik Krallık ‘ta EBYS ticari hacmi Şekil 1.1’de gösterilmektedir.



Şekil 1.1. Birleşik Krallık ‘ta EBYS ticari hacim. [16]

Global Market Insights, Inc. tarafından hazırlanan yeni bir araştırmaya göre; bulut bilişimin benimsenmesi, BYOD trendinin, Software as a Service (SaaS) tabanlı çözümlerin ve big datanın kullanımının artmasının, elektronik belge yönetim sistemi pazarının büyümesini teşvik etmesi beklenmektedir. Bu tür çözümler, kuruluşların iş akışını düzene koymalarını ve dokümantasyon ihtiyaçlarını etkin bir şekilde yönetmelerini, böylece bireysel üretkenliği geliştirmelerini ve kuruluşların yapılandırılmamış içeriği merkezileştirmelerini ve işle ilgili bilgileri hızlı ve etkin bir şekilde çözmelerini sağlayacaktır. EBYS, belgenin elektronik olarak oluşturulmasını, çoğaltılmasını, silinmesini ve dağıtımını kontrol eder. Ayrıca, aynı belgenin birden

çok sürümünü kontrol etmeye yardımcı olur ve birleştirilmiş aramalar yapılması sağlar [17].

Rapora göre, sektörler arasında veri uyumluluğunu sürdürme ihtiyacı gerektiren düzenlemelerin, EBYS pazarının büyümesini teşvik etmesi beklenmektedir. Ancak, çevrimiçi belge yönetim sisteminin siber güvenlik sorunlarının olması ve şirket içi çözümlerin yüksek uygulama ve bakım maliyetinin bulunması bu büyümeyi engelleyebilecek önemli faktörler olacağı belirtilmiştir. Büyümeyi etkileyen ana trendler arasında sosyal entegrasyon, iş akışı iş birliği, özelleştirilmiş müşteri portalları ve mobil cihaz entegrasyonu yer almaktadır. EBYS pazar çözümleri, belgelerin arşivlenmesini, indekslenmesini, erişilmesini ve geri alınmasını sağlamaktadır [16].

Küresel düzeyde tüm alanlarda ortaya çıkan veri hacminin artmasıyla yeni sorunlar ortaya çıkmıştır. Üretilen büyük veriyi depolamak, yönetmek ve analiz etmek bu sorunların başında gelmektedir. Ayrıca, verilerin depolanması, üretilmesinden daha pahalı ve zor hale gelmiştir [18]. Bu şekilde, oluşturulan, depolanan ve yönlendirilen veri miktarı önemli ölçüde artmış ve bu artış nedeniyle, “Big Data” ve veri bilimi / bilgisi gelişmeye başlamıştır [19].

1.1. Literatür

Big data, yalnızca bir bilgisayarda depolanamayan ve yönetilmeyen verilerdir. Günümüzde, big datayı yönetebilmek için, bilgisayarlar dağıtık bir dosya sistemine bağlanıyor ve ağda birlikte çalışıyorlar. Dağıtık Dosya Sistemleri, node (düğüm)'lardan oluşan kümelere ayrılır. Performans, veri güvenliği, ölçeklenebilirlik, kullanılabilirlik, kolay erişilebilirlik, sağlamlık ve güvenilirlik, büyük verilerin en önemli özellikleridir. Big data yönetimi problemi, Dağıtık Dosya Sistemi ve ağ altyapısı kullanılarak çözülür.

Dağıtık Dosya Sistemi ile ilgili çalışmalar 1970'lerde başlamıştır [20]. İlk çalışmalardan biri ROE'dir ve replica tutarlılığı, kolay kurulum, güvenli dosya yetkilendirme ve ağ şeffaflığı özellikleri öne çıkan özelliklerdir. [21].

1981'de geliştirilen LOCUS, ağ şeffaflığı, yüksek performans ve yüksek güvenilirlik özellikleri sunan bir başka Dağıtık Dosya Sistemidir [22]. Network File System (NFS), 1984'te Sun Microsystems tarafından geliştirilmeye başlanmıştır. Bu sistem, UNIX'te en çok kullanılan Dağıtık Dosya Sistemidir. Remote Procedure Call (RPC) iletişim için kullanılır [23]. Unix dosya sisteminin "dağıtık" bir şekilde çalışmasını sağlamak için tasarlanmıştır. Sanal dosya sistemi bir katman olarak kullanılır. Bu nedenle, istemciler farklı dosya sistemlerini kolayca çalıştırabilir ve NFS'de hata toleransı yüksektir. Dosya durumu bilgileri tutulur ve bir hata oluştuğunda, istemci bu hata durumunu hemen sunucuya rapor verir. NFS'de dosya replikasyonu yapılmaz, tüm sistem çoğaltılır [24]. Sadece dosya sistemi NFS'de paylaşılır, yazıcı veya modem paylaşılamaz. Paylaşılan nesneler, bir dizinin yanı sıra bir dosya da olabilir. NFS'de yerel bir disk üzerindeki her uygulamayı ayarlamak gerekmez ve sunucu kullanılarak paylaşılabilir. Her şey için aynı bilgisayar hem sunucu hem de istemci olabilir. Sonuç olarak, NFS veri depolama maliyetlerini azaltır.

Andrew Dosya Sistemi (AFS-1983) ve onun alt versiyonları CODA (1992) ve OpenAFS [25], dağıtık dosya sistemleri için open source (açık kaynak) uygulamalardır. Bu sistemler ölçeklenebilir ve daha büyük cluster boyutlarına sahiptir. Ayrıca, sunucu yükünü azaltırlar ve tüm dosyayı önbelleğe alabilirler. CODA, erişilebilirliği artırmak için birden çok sunucu da replikasyon yapar. AFS sadece Unix, OpenAFS ve CODA desteği MacOS ve Microsoft Windows'u desteklemektedir. Bu sistemlerde, tüm istemciler için aynı namespace oluşturulur. Ancak, replikasyon sınırlıdır ve bunun için Read-one / Write-all (ROWA) şeması kullanılır [26,27].

Frangipani, 1997 yılında iki katmanlı olarak tasarlanan yeni bir dağıtık dosya sistemi olarak geliştirilmiştir. Alt katman, sanal disklerden oluşmaktadır. Bunlar depolama hizmetleri sağlar. Otomatik olarak ölçeklendirilebilmekte ve yönetilebilmektedir. Üst

katmanda, Frangipani dosya sistemini kullanan birden fazla makine var. Bu makineler paylaşılan sanal diskte dağıtık olarak çalışırlar. Frangipani dosya sistemi, tutarlı ve aynı dosya grubuna paylaşılmış erişim sağlar. Sistemde kullanılan veriler büyüdükçe, daha fazla depolama alanı ve daha yüksek performanslı donanım elemanları gereklidir. Sistem bileşenlerinden biri çalışmıyorsa, diğerinin kullanılabilirliği devam ettiğinden sistem hizmet vermeye devam eder. Sistem büyüdükçe, eklenen bileşenler yönetimi karmaşıklaşmaz ve dolayısıyla insan yönetimine daha az ihtiyaç duyar [28].

FARSITE (2002), bir ağ üzerinde dağıtık olarak çalışan sunucusuz bir dosya sistemidir. Fiziksel olarak güvenilmez bilgisayarlardan oluşan bir ağ üzerinde dağıtılmış çalışır. Sistem sunucusuz, dağıtık dosya sistemidir. Merkezi bir yönetime ihtiyaç duymaz. Böylece, sunucu sistemi personeli gibi maliyeti yoktur. FARSITE, bir masaüstü bilgisayarın I/O iş yükünü üniversite veya büyük bir şirkette desteklemek için tasarlanmıştır. İstemciyi önbelleğe alma yoluyla yeterli bir performans, hazır bulunma, replikasyon yoluyla erişilebilirlik ve hazır bulunma, şifreleme kullanarak kimlik doğrulama, ad alanı temsilci kullanarak ölçeklenebilirlik özelliklerini sağlamaktadır. FARSITE'nin en önemli tasarım hedeflerinden biri, Byzantine hata toleransının faydalarını kullanmaktır [29].

2006 yılında açıklanan CEPH dosya sistemi, nesne depolama yapan benzer sistemlerin üst katmanında yer almaktadır. Bu katman, verileri ve meta data yönetimini ayırır. Bu, güvenilmez Object Storage Devices (OSDs) için tasarlanmış rastgele veri dağıtım işlevi (CRUSH) ile gerçekleştirilir. Bu işlev dosya ayırma tablosu (FAT)'nun yerini alır. CEPH ile dağıtık veri replikasyonu, hata algılama ve nesne depolama cihazlarına aktarılmış kurtarma işlemleri, yerel dosya sisteminde çalışmaktadır. Böylece sistem performansının arttırılması hedeflenmiştir. Dağıtık bir meta data seti, yönetimi son derece verimli hale getirmiştir. Reliable Autonomic Distributed Object Store (RADOS) katmanı tüm dosyalama süreçlerini yönetir. Ölçümler, farklı disk boyutlarında da çalışabilen CEPH'in performansını test etmek için çeşitli iş yükleri altında gerçekleştirilmiştir. Sonuç olarak, I/O performansı oldukça yüksek çıkmıştır. Ölçeklenebilir bir meta data yönetimine sahip olduğu

görülmüştür. Ölçümlerden dolayı saniyede 250.000 meta işlemi desteklediği görülmüştür. CEPH ile yüksek performanslı, güvenilir ve ölçeklenebilir dağıtık dosya sistemi geliştirilmiştir [30].

2007 yılında Hadoop, Hadoop Dağıtılmış Dosya Sistemi (HDFS) ve MapReduce-paralel hesaplama aracı kullanılarak geliştirilmiştir. Hadoop, çok büyük veri kümelerinin analizini ve dönüşümünü sağlayan bir frameworktur. HDFS, big datayı, standart sunucularda kümelere bölerek dağıtır. Veri güvenliğini sağlamak için sunuculara blokları kopyalayarak yedekler [31]. Hadoop / MapReduce, big datayı verileri işlemek ve yönetmek için kullanılır. Map fonksiyonu, verileri kümeye dağıtır ve işlenmesini sağlar. Reduce işlevi, verilerin birleştirilmesini sağlar. Hadoop'un ölçeklenebilirlik özelliğine sahiptir ve veri PB seviyesinde veriyi kolayca kullanılabilir [32]. Bugün, Hadoop birçok büyük şirket tarafından kullanılmaktadır. Özellikle, endüstriyel ve akademik alanlarda tercih edilir. LinkedIn, eBay, AOL, Alibaba, Yahoo, Facebook, Adobe, IBM gibi şirketler genellikle Hadoop'u kullanmaktadır [33].

2015'te duyurulan CalvinFS, meta data yönetimi için yüksek verimliliğe sahip olarak tasarlanmış bir veri tabanı kullanarak replica özelliğine sahip, ölçeklenebilir bir dosya sistemidir. Bunun için meta dataları, birden çok node'a yatay olarak ayırır. Meta data öğesini düzenlemesi gereken dosya işlemleri dağıtık olarak çalışır. Bu sistem ayrıca standart dosya sistemlerini de destekler. Bu dosya sistemi yaklaşımı, ölçeklemenin milyarlarca dosyaya kadar yapılabileceğini göstermiştir. Okuma gecikmelerini azaltırken, aynı anda saniyede yüz binlerce güncelleme ve milyonlarca okuma işlemi yapabilir [34].

2016 yılında Mohammed S. Al-Kahtani ve Lutbul Karim, ölçeklendirilebilir bir dağıtık sistem frameworku sundu [35]. Sistem merkezi sunucuda ölçekleme gerçekleştirmektedir. Önerilen frameworkte veri toplama işleminde, toplanan veri miktarı arttıkça işlem sunucu tarafından diğer bilgisayarlara aktarılır. Diğer bir deyişle, sistem veri miktarı arttığında dağıtık çalışır. Bu çalışmaya benzer diğer dağıtık frameworkler şunlardır: IdeaGraph algoritması [36], olasılıksal latent

semantik analiz (PLSA) [37], yerel farkında zamanlama algoritması [38], en yakın komşu algoritması [39], nesne tabanlı ortak çalışan filtreleme algoritması [40], öneri algoritması [41], konveks optimizasyon [41] ve paralel iki geçişli MDL (PTP-MDL) [41].

Erguzen ve Erdal tarafından sağlık hizmetlerinde big data ile ilgili yapılan çalışmada, Magnetic Resonance Imaging (MRI) üzerinde Region of Interest (ROI)'i depolamak için yeni bir dosya yapısı ve arşivleme sistemi geliştirilmiştir. Sistemde, MRI'dan hastayla ilgili hayati bilgiye sahip olan ROI kısmının dışında kalan alanları atıp, ROI bölgesinin sıkıştırılarak depolanmasını sağlayarak %30 daha az yer tutan bir dosya sistemi geliştirmişlerdir. Ancak, bu çalışma sadece görüntü boyutunu azaltmış, Dağıtık Dosya Sistemi üzerinde big dataları etkin bir şekilde saklamak için tasarlanmamıştır [11].

Yang Jin ve diğerleri, elektronik sağlık kayıtları için dağıtık bir depolama tablosu tasarlamışlardır. Sistem, Hadoop DFS-MapReduce frameworkunde çalışan dağıtık sütun tabanlı veri tabanı olan H-Base'de tasarlanmıştır. Model düşük maliyetli ve iki name node'a sahiptir. Ayrıca, daha iyi bir performans elde edebilmek için depolama yük dengesini sağlayan HMaster ve HRegionServer kullanılmaktadır. Bununla birlikte, sistemin HDFS için veri blok çoğaltma stratejileri geliştirmesi için çalışması yapması gerektiği belirtilmiştir [42].

Günümüzde dağıtılmış dosya sistemleri iki ana kategoride ele alınabilir:

- Big data depolama: Big dataları tutmak için gerekli dosya sistemi ve dizin şemasını kullanır (veri kaydetme).

- Big data analizi: Grid computing araçları ile node'larda toplanan verilerin en kısa ve tutarlı analizi (veri madenciliği).

Thomas Ryana ve diğ. [43], Multi-Tier Resource Allocation (MTRA)-Çok Katlı Kaynak Ayırma tekniğini, big datanın dağıtık sistemler için yeni bir kaynak yönetim

tekniği olarak önermişlerdir. Bu tekniğin verimliliğinin, veri bakımından yoğun bir bilgi işlem ortamında, Hadoop YARN'de MapReduce veri işleme frameworku ile test edildiğini ve rapor edilen sonuçlara göre, MTRA tekniğinin, Hadoop YARN ortamında bir dizi MapReduce testinde performansının %18'e kadar arttığını belirtmişlerdir. Facebook iş yükü modelinde, iş yürütme süresinin ortalama %10, bireysel işlerde %56'ye varan oranlarda geliştiğini gösterilmiştir. Bu, MTRA'nın özellikle bilgi işlem ortamları için etkili bir genel amaçlı kaynak yönetim tekniği olduğunu göstermektedir.

Miguel A. Martínez-Prieto, Carlos E. Cuesta, Mario Arias, Javier D. Fernández [44], gerçek zamanlı yönetim için Solid mimarisini ortaya koymuşlardır. Big datayı yönetmenin bu yönteminde, yüksek veri hızı ve büyük hacimli veriler vardır. Bu parametreler gereklidir çünkü veri gün geçtikçe artmaktadır. Ve bu veri, verilerin doğru yönetimi olmadan ele alınamaz. Ancak bu mimari, big datanın gerçek zamanlı yönetiminde yardımcı olabilmektedir.

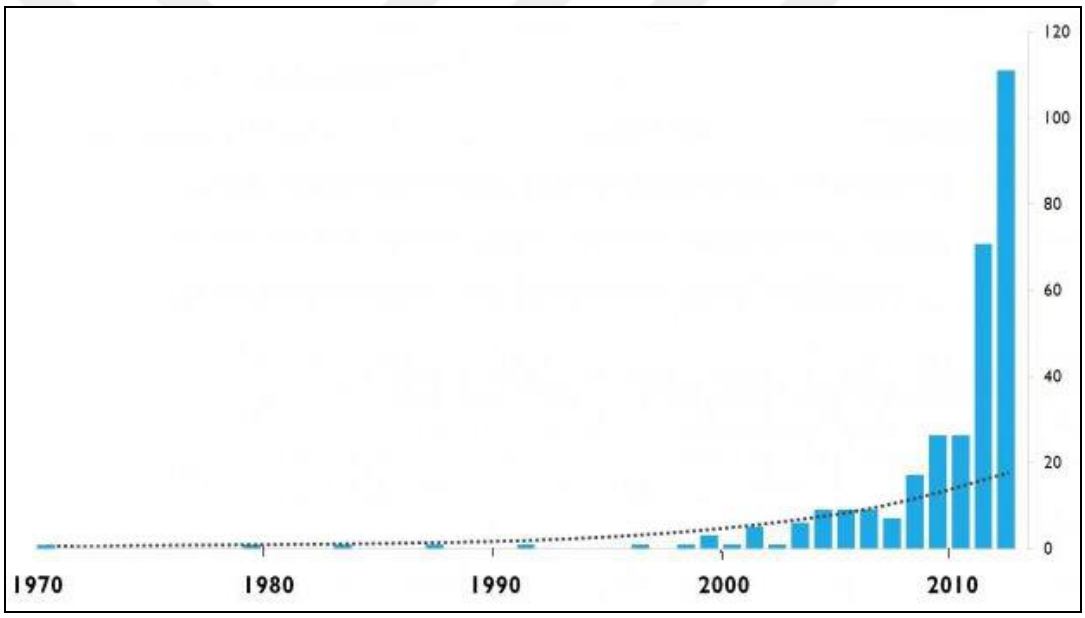
Boxwood projesi [45], dağıtık olan kabul edilen, kilitleme, dağıtık yığın depolama ve dağıtık B-tree depolama alanı sağladığından, Chubby, Google File System (GFS) ve Bigtable ile bir şekilde örtüşen bileşenlere sahiptir. Her durumda, Boxwood'un bileşeninin, ilgili Google hizmetinden biraz daha düşük bir seviyeyi hedeflendiği anlaşılmaktadır. Boxwood projesinin amacı, sistem veya veri tabanı gibi üst düzey hizmetler oluşturmak için altyapı sağlamaktır; Bigtable'ın hedefi ise, verileri depolamak ve isteyen istemci uygulamalarını doğrudan desteklemektir.

Abhaya Kumar Sahoo ve arkadaşları tarafından önerilen algoritma [46], büyük XML belgelerini ayrıştırmak ve aramak için MapReduce yaklaşımına ve GPU'ya dayanmaktadır. Önerilen algoritmanın XML endeksleme alanlarında daha iyi bilgi işlem hızı sağladığı belirtilmiştir. Bu çalışmada NVIDIA'nın JCUDA programlama ortamına sahip olan GPUsu, paralel hesaplama paradigmasını sağlamak için kullanılmıştır. Hadoop frameworku ile GPU'ların yüksek performansının bütünleştirilmesi, doku eşleşmesi için Boyer-Mores genişletilmiş algoritmasının uygulanmasından sonra gerçek dünyadaki veriler ve dosyalar üzerinde uygulanmıştır.

Hadoop'un verinin artmasıyla benchmarking sistemiyle farklı algoritmalar için veri işleminde harcadığı süreyi hesaplamışlar ve 3,5 kat hız artışı olduğunu gözlemlemişlerdir.

1.2. Büyük Veri (Big Data)

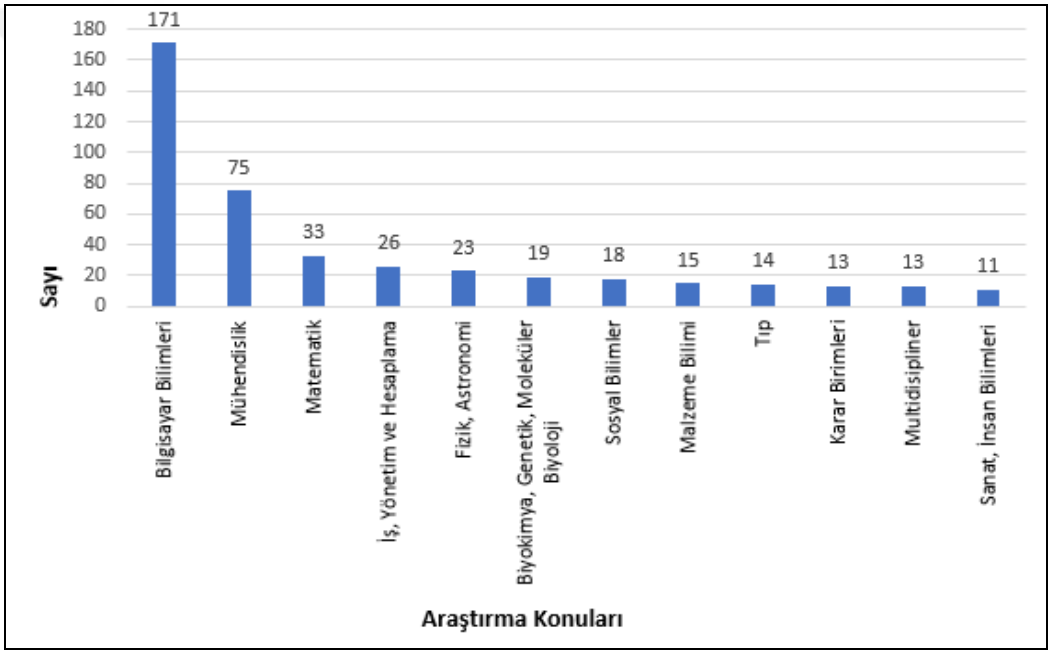
Big data, klasik veri işleme yöntemleriyle işlenemeyecek büyüklükteki veri kümesi olarak tanımlanabilir. İlk "Big Data" terimi, 2005 yılında, O'Reilly medyasından Roger Magoulas tarafından kullanılmıştır [47].



Şekil 1.2. Big data yayınlarının yıllara dağılımı

Son zamanlarda, endüstriyel alanlar, ticaret, eğitim, sağlık, devlet kurumları işlemleri, astronomi, bankacılık ve borsa gibi sektörlerin ürettiği veriler sayesinde artık sistemlerin verileri big data olarak isimlendirilebilmektedir. G. Halevi ve H. F. Moed'nin 2012 yılında yaptığı çalışmada, big data ile ilgili Scopus'ta benzer makaleler incelenmiş ve ortaya çıkan veriler tartışılmıştır. Araştırılan makaleler

hakemli bilimsel makalelerdir. Çalışma sonucunda, big data ile ilgili makalelerin türü, yayınlanma zamanları gibi konularda sonuçlar elde edilmiştir. Şekil 1.2'de yayın sayısı görülmektedir. 2008'den günümüze big data ile ilgili araştırma makalelerinin artması, son on yılda konuya daha fazla ilgi gösterildiği için kolaylıkla açıklanabilir. Ancak, terimin kullanıldığı eski örneklere daha yakından bakıldığında, örneğin, big data teriminin ilk geçtiği makalenin, atmosfer ve okyanus sondajlarıyla ilgili 1970 tarihli bir makale olduğu görülmüştür. Bu makalede okyanusla ilgili ortaya çıkarılan verilerin o gün için büyük olduğu düşünülmüş ve veriyi işlemek için kullanılan bilgisayarın fotoğrafı kullanılmıştır [48].

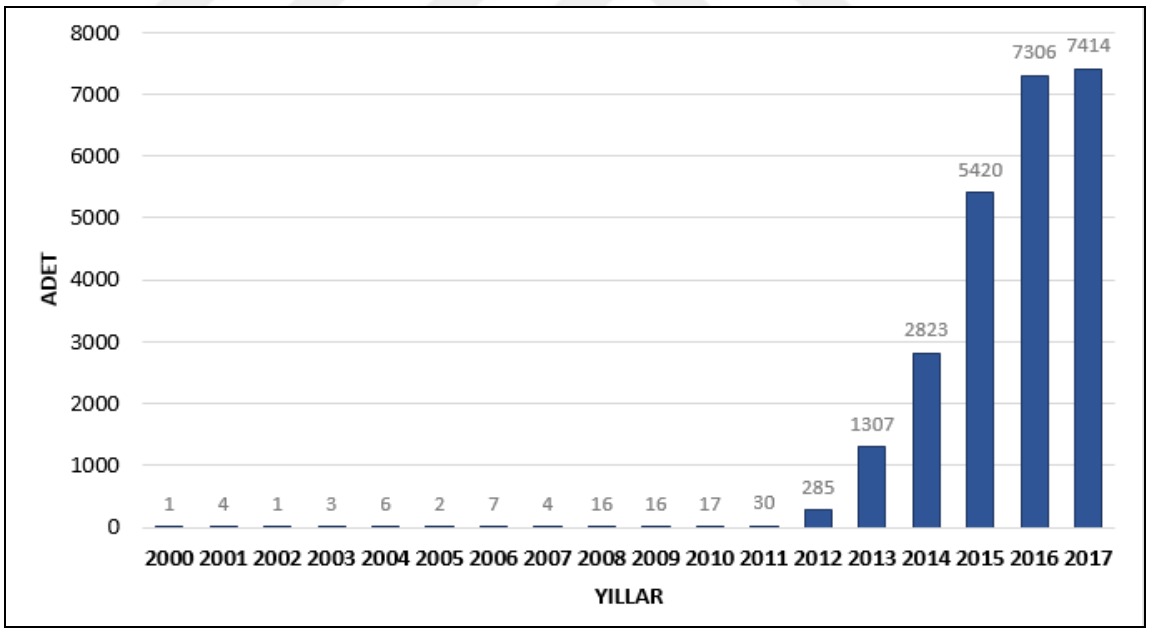


Şekil 1.3. Konulara göre big data araştırmaları.

Terimin kullanıldığı diğer erken dönem makaleleri ise genellikle dilbilim, coğrafya ve mühendislik gibi alanlarda büyük veri kümeleri için bilgisayar modelleme ve yazılım/donanım geliştirme konuları ile ilgilidir. Yayınlanan makalelerin konu istatistiğinin gösterildiği Şekil 1.3 incelendiğinde, Gali Halevi ve diğerlerinin yazdığı makale [49]'ye göre, 2000 yılına kadar yapılan ilk yayınların, özellikle bilgisayar

mühendisliği (yapay sinir ağları, yapay zekâ, bilgisayar simülasyonu, veri yönetimi) başta olmak üzere, mühendislik alanlarında yoğunlaştığını görmekteyiz. 2000 yılından itibaren ise, bilgisayar bilimleri ve matematik alanında çalışmalar yoğunlaşmıştır.

Yayınların türleri açısından bir başka ilginç bulgu, konferans yayınlarının dergi makalelerine göre daha çok olmasıdır. 2012 yılında Scopus'un endekslediği yayınlarda konferans bildirilerinin araştırma makalelerinden iki kat fazla olduğu görülmektedir. Bilim alanı analizinde ise, bilgisayar bilimi ilk sırada yer almış daha sonra mühendislik, matematik, işletme ve ayrıca sosyal ve karar bilimleri sıralaması şeklinde devam etmiştir [49]. Big datayı oluşturan en önemli faktör, internetin hızla yaygınlaşarak kullanımının artması ve internete bağlı cihaz sayısının artmış olmasıdır. 2003 yılında dünyada kişi başına düşen birbirleriyle bağlantılı cihaz oranı 0,08 iken bu oranın 2020 tahmini ise 6,48'dir [50].



Şekil 1.4. 2000-2017 yılları "Big Data" konulu Web of Science yayın sayısı [51].

Dünya'da Big data konusundaki çalışmalarda hızla artmaktadır. Web of Science akademik veri tabanında konusu big data olan çalışma sayısı Şekil 1.4'te gösterilmiştir. Şekle göre 2000-2017 yılları arasında yapılan çalışmaların 26 kat arttığı görülmüştür. Aynı şekilde, Google arama motorunda Mart-2013'ten Ocak-2017'arasında big data aramalarının iki katına çıktığı görülmüştür [52].

2015'te A. Gandomi ve M. Haider'in yazdığı makalede [53], big datanın tanımı ile ilgili çalışma yapmıştır. Laney, 2001'de yazdığı yazıda, volume, variety ve velocity (3V) veri yönetimindeki zorlukların üç boyutu olduğunu öne sürmüştür [54]. 3V, big data'yı tanımlamak için ortak bir framework olarak ortaya çıkmıştır [55]. Gartner, Inc. Big data'yı benzer terimlerle tanımlar: "Big data, gelişmiş öngörü ve karar alma için bilgiyi işlemenin yenilikçi şekillerini maliyet-etkin bir şekilde ortaya koyan yüksek hacimli, yüksek hızlı ve çok çeşitli bilgi varlıklarıdır" [56].

Benzer şekilde, TechAmerica Vakfı büyük verileri şu şekilde tanımlar: "Big data, bilginin üretilmesi, depolanması, dağıtımı, yönetimi ve analizi için gelişmiş teknikler ve teknolojilere ihtiyaç duyan yüksek hızlı, karmaşık ve değişken büyük hacimleri tanımlayan bir terimdir" [57].

1.2.1. Big Data Veri Türleri

Dünya üzerinde üretilmiş olan veriler, türlerine göre üç yapıda sınıflandırılabilmektedir. Bunlar, yapılandırılmış veriler, yapılandırılmamış veriler ve yarı yapılandırılmış veri türleridir.

Yapılandırılmış veriler, yüksek bir organizasyon derecesine sahip bilgileri ifade eder, öyle ki ilişkisel bir veri tabanına dahil olmak, basit arama motoru algoritmaları veya diğer arama işlemleri ile sorunsuz ve kolayca aranabilir verilerdir; yapılandırılmamış veriler ise tam tersi şeklinde kolayca arama motorları tarafından aranamazlar.

Yapı eksikliği, derlemeyi zaman ve enerji tüketen bir iş haline getirir. Yapılandırılmamış verilerin kuruluşa eklediği doğal maliyetleri azaltmak için bir veri analizi mekanizması bulmak, tüm iş katmanlarındaki bir şirket için yararlı olmaktadır [58].

Elbette, yapılandırılmamış verileri yapılandırılmış verilere dönüştürmek mümkün olsaydı, o zaman bu işlem için yapılan tüm organizasyonel işlemler ortadan kalkabilirdi; bununla birlikte, yapılandırılmış veriler makine dili ile ilgilidir, çünkü bilgisayarların kullanımıyla ilgili bilgileri daha kolay hale getirir; yapılandırılmamış veriler, genellikle insanlar için daha uygundur, veri tabanı biçimindeki bilgi ile kolayca etkileşime girmeyen bir veri türüdür.

E-posta, yapılandırılmamış verilerin için bir örnektir; çünkü kurumsal bir insan kaynakları yöneticisinin yoğun kullandığı gelen posta kutusu tarih, saat veya boyuta göre düzenlenebilir; eğer gerçekten tamamen yapılandırılmışsa, aynı zamanda sapma veya yayılma olmaksızın tam konu ve içerik bakımından düzenlenebilirdi. Çünkü insanlar genellikle odaklanmış e-postalarda bile tam olarak bir konu hakkında yazışmazlar [58].

Yarı yapılandırılmış veriler, ilişkisel veri tabanları veya diğer veri tabloları ile ilişkili veri modellerinin biçimsel yapısına uymayan yapılandırılmış bir veri şeklidir. Ancak yine de anlamsal öğeleri birbirinden ayırmak için etiketler veya başka işaretleyiciler içerir ve verilerdeki kayıtların ve alanların hiyerarşilerini zorlar. Bu nedenle, aynı zamanda şematik veya kendi kendini tanımlayan bir yapı olarak bilinir.

Öte yandan e-tablolar, ilişkisel veri sistemi olarak değerlendirileceğinden bilgi için hızlıca taranabilen yapılandırılmış veriler olarak kabul edilir. Yapılandırılmamış verilerin ortaya çıkardığı problem hacimdir. Şimdiye kadarki en çok iş etkileşimi bu türden olup, web tabanlı bir arama motorunda olduğu gibi, gerekli öğeleri elemek ve çıkarmak için büyük bir kaynak yatırımı gerektirmektedir [58].

Veri havuzu çok büyük olduğu için, mevcut veri madenciliği teknikleri genellikle, buradaki bilgilerin önemli bir kısmını kaçırmaktadır. Eğer bunlar etkin bir şekilde analiz edilebilseydi, bu verilerin çoğu, çok iyi bir şekilde analiz sonucunu değiştirebilecek veriler olabilirdi.

Veri yönetimi dünyasındaki yaygın eğilim, düz dosya veri kümelerinden, veri bütünlüğünün giriş, çıkış, depolama, sorgulama ve bakım işlemlerini kolaylaştıran bilgisayar yönetim sistemlerinde yönetilen, çok tablolu veri tabanlarına doğru bir harekettir [59].

Yapılandırılmış veri türleri, veri tabanı dosyasının tabloları, yarı yapılandırılmış veriler, XML dosyaları, sosyal medya verileri, yapılandırılmamış veriler ise, algılayıcılardan gelen veriler, e-postalar gibi verilerdir.

1.2.2. Big Datanın Kavramları

Big data, üç kavramdan oluşur; volume, velocity ve variety. Elektronik belge kayıtları ile ilgili olarak yapılan big data uygulamalarının da bunları kapsaması gerektiği kabul edilir.

Volume, verilerin büyüklüğünü ifade eder. Big data boyutları birden fazla TB ve PB olarak sınıflandırılır. IBM tarafından 2012 yılının ortasında yapılan bir ankette, 1144 ilgilinin yarısından fazlasının bir TB'tan büyük veri olmak üzere veri kümelerini dikkate aldığını ortaya çıkarmıştır [60]. Bir TB, 1500 CD veya 220 DVD'ye sığacak kadar yaklaşık 16 milyon Facebook fotoğrafını saklayabilecek big datayı depolar. Big data hacimlerinin boyutu görecelidir, zaman ve veri türü gibi faktörlere göre değişmektedir. Bugün açıklanan veriler, gelecekte daha big data kümelerinin depolanmasına olanak tanıyarak, ileriki dönemlerdeki big data eşik değerlerini karşılayamayabilir. Ayrıca, variety konusu altında ele alınan veri türü, "Big" ile ne ifade edildiğini tanımlar. Aynı boyuttaki iki veri kümesi, türlerine bağlı olarak (video, tablo, metin) farklı veri yönetimi teknolojilerini gerektirebilir. Ayrıca, big

datanın tanımları endüstriye bağlıdır. Bu hususlar, big datanın boyutunu belirlemenin pratik olmadığını göstermektedir [53]. Büyük miktarda bilgiyi işleme yeteneğinden kazanılan fayda, big data analitiğinin ana konusudur. Daha iyi ürün ortaya çıkarabilmek için daha çok veri ile çalışmak şirketlerin temel prensiplerindendir. Bunun sonucu olarak pek çok şirket, çok çeşitli miktarda veri depolamaya meyilli olmuştur: sosyal ağ verileri, sağlık verileri, finansal veriler, biyokimya ve genetik veriler, astronomik veriler, eğitim verileri, elektronik belge verileri bunlar arasındadır [61].

Variety, bir veri kümesindeki yapısal çeşitliliği ifade eder. Teknolojik gelişmeler, firmaların çeşitli yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış veriyi kullanmalarına izin verir. Mevcut verilerin sadece %5'ini oluşturan yapılandırılmış veriler [62], elektronik tablolarda veya ilişkisel veri tabanlarında bulunan tablo verileri olarak tutulmaktadır. Metin, resim, ses ve video, bazen analiz edilmek için makineler tarafından gerekli olan yapısal organizasyondan yoksun olan, yapılandırılmamış verilerin örnekleridir. Tamamen yapılandırılmış ve yapılandırılmamış veriler arasında bir süreklilik içeren, yarı yapılandırılmış veri formatı katı standartlara uymamaktadır. Web üzerinde veri değişimi için bir metin dili olan Extensible Markup Language (XML), yarı yapılandırılmış verilerin tipik bir örneğidir. XML belgeleri, kullanıcı tarafından okunabilir hale getiren kullanıcı tanımlı veri etiketleri içerir. Big datanın karakteristik bir özelliği olan variety, yeni bir özellik değildir. Kuruluşlar, dahili kaynaklardan (örneğin, sensör verileri) ve harici kaynaklardan (ör., sosyal medya) ortay çıkarttıkları yapılandırılmamış verileri depolamaktadırlar. Bununla birlikte, organizasyonların iş süreçlerinde veri kullanmasına olanak tanıyan yeni veri yönetimi teknolojilerinin ve analitiğinin ortaya çıkışı, yenilikçilik açısından önemlidir. Örneğin, yüz tanıma teknolojileri, işyerinin müşteri trafiğini, müşterilerinin yaş ve cinsiyetini, onların mağaza içinde hareket yerleri hakkında bilgi sahibi olmasını sağlar. Bu paha biçilmez ürün promosyonları, yerleştirme ve personel ile ilgili kararlar almasında yardımcı olurlar. Tıklama verileri, müşteri davranışları ve online perakende satıcılarına gezinme durumları hakkında ok sayıda bilgi sağlar. Tıklama veri akışı, bir müşteri tarafından görüntülenen sayfaların zamanlaması ve sıralaması konusunda önerilerde bulunur.

Big data analitiği kullanarak, küçük ve orta ölçekli işletmelerde web sitesi tasarımlarını geliştirmek ve etkili kişiselleştirilmiş ürün satışı ve önermesi uygulayabilmek için büyük miktarlarda yarı yapılandırılmış veriyi kullanabilmektedirler [53]. Bu veriler sabit bir yapıya sahip değildir ve nadiren kendilerini düzenli bir biçimde ve işlenmeye hazır halde sunarlar [63]. Gerçekten de ilişkisel veri tabanlarından gelen yapılandırılmış veriler, web günlükleri, sosyal medya gönderileri, doğrudan bir sensör kaynağından gelen veriler, e-postalar gibi yarı yapılandırılmış veriler veya video, hareketsiz görüntüler, ses, tıklamalar gibi yapılandırılmamış veriler olabilir [64].

Velocity: Big data sürekli bir şekilde hızla büyümektedir. Kısa sürede çok daha büyük boyutlara ulaşacaktır. Big datayı kullanan ve işleyen donanım ve yazılımların da bu büyümeye cevap verebilecek niteliklerde olması gerekmektedir. Aynı şekilde, bu veriyi işleyecek işlem hazınında geliştirilmesi gerekmektedir. Velocity, verinin hangi oranda üretildiğini ve hangi hızda analiz edileceğini gösteren bir kavramdır. Akıllı telefonlar ve sensörler gibi dijital cihazların çoğalması, daha önce görülmemiş bir verinin ortaya çıkma hızına yol açtı ve bu, gittikçe artan gerçek zamanlı analize ve doğrulamaya dayalı planlama ihtiyacını ortaya çıkarıyor. Klasik ticari şirketler bile yüksek frekanslı veri üretiyorlar. Örneğin Wal-Mart, saatte bir milyondan fazla işlem gerçekleştirmektedir [62]. Mobil cihazlardan çıkan ve mobil uygulamalardan akan veriler, kurumlar için gerçek zamanlı, kişiselleştirilmiş değerler üretmek için kullanılabilecek bilgi akışları üretir. Bu veriler, gerçek zamanlı olarak analiz edilebilen coğrafi konum, demografi ve geçmiş satın alma modelleri gibi sağlam bilgiler sağlamaktadır [53]. Velocity, veri akışları, yapılandırılmış kayıt oluşturma ve erişim ve teslimat için hazır bulundurulmayı kapsar. Gerçekten de bu, sadece gelen verilerin konusu değildir: Hızlı hareket eden verileri daha sonraki toplu işlem için yığın depolama alanına yönlendirmek mümkündür. Daha da önemlisi, geri besleme döngüsünün hızıdır. Bu da big data işlemleri için önemli bir unsurdur [63]. Bu üç V'ye ilave olarak, big data verilerin diğer boyutları da kaynaklarda açıklanmıştır.

Bunlar:

Value: Oracle, big datayı tanımlayan bir özellik olarak Value'yi tanıtmıştır. Oracle'ın tanımına göre, big datalar genellikle nispeten “düşük değer yoğunluğu” ile karakterize edilmektedir. Yani, orijinal formda alınan veriler genellikle hacmine göre düşük bir değere sahiptir. Ancak, büyük miktardaki veriyi analiz ederek yüksek bir değer elde edilebilir. Big datayı işleme ve yönetimi sonucunda ortaya çıkan veriler, sisteme ilave değer katmalıdır [53].

Veracity: Big data içindeki kayıtların düzenli olması, karmaşık olmaması, basit olması ve kompleks olmaması gerektiğini gösterir. Örneğin personel takibi yapan bir sistemde, personelin girişlerini takip eden sensörün gönderdiği verinin yanlış olması, farklı formatta olması veya gelen tarih bilgisinin 1800’lü yıllar olması bilginin doğru olmadığını gösterir. İlave olarak verinin belirsiz olması da big data işlemleri için bir sorundur.

Çizelge 1.1. Big data verilerinin kaynak türleri

Sektör	E-ticaret	Yüzde Değer
İş ve Ekonomi	Evet	24
Bilişim Hizmetleri	Evet	16
İletişim Ve Medya	Evet	6
Eğlence	Evet	1
Spor ve Rekreasyon	Evet	2
Sağlık	Hayır	4
Kamu	Hayır	2
Yerel	Hayır	4
Sosyal ve Kültür	Hayır	9
Eğitim	Hayır	16
Sanat ve Toplum	Hayır	4
Bilim	Hayır	2
Başvuru	Hayır	8
Diğerleri	Hayır	2

Bu konuyu göz önüne alarak veracity'yi IBM, bazı veri kaynaklarının doğasında var olan güvenilmezliği temsil eden dördüncü V olarak ele almıştır. Örneğin, sosyal medyadaki müşteri görüşleri, insan kararı ile oluştuğu için görecelidir ve doğası gereği belirsizdir. Yine de değerli bilgiler içerirler. Dolayısıyla, kesin olmayan ve belirsiz verilerle başa çıkma ihtiyacı, belirsiz verilerin yönetimi ve madenciliği için geliştirilen araçlar ve analizler kullanılarak ele alınan big datanın bir başka yüzüdür [53].

Variability: Big datadaki verilerin farklı kaynaklardan gelmesi nedeniyle türlerinin farklı olmasını gösteren kavramdır. Deneysel olarak, yapılandırılmış verilerin bazı alanlarda hacim ve kapsama özellikleri Çizelge 1.1'de gösterilmiştir [65]. Veriler, metin, ses, video, resim, web sayfası, e-posta formatında olabilir.

Bu verilerin birbirleriyle dönüştürülmesi, birleştirilmesi ve eşleştirilmesi gerekebilir. Big data bu karmaşık verilerin yönetimini hızlı ve doğru bir şekilde yapabilir [63].

1.2.3. Big Datanın Zorlukları

Bu bölümde big datanın yönetiminde karşımıza çıkan problemlere yer verilmiştir.
Big data analizi, veri toplama ve kaydetme, bilgi çıkarma ve temizleme, veri entegrasyonu, toplama ve temsil, sorgulama, veri modelleme ve analiz ve yorumlamayı içeren çoklu farklı aşamaları içerir.

Bu aşamaların her biri farklı problemler, zorluklar ortaya çıkartmaktadır; Verilerin heterojen olması, ölçeklenebilirlik, zamansallık, karmaşıklık ve gizlilik big data madenciliğinin bazı zorluklarıdır.

Bu zorluklar aşağıda verilmiştir:

1. Big data analizinin zorlukları, toplanan ve depolanan verilerdeki farklı modellere veya kurallara (karışık veriler) dayanan karma verilerin var olmasının yanı sıra,

büyük ölçeğinden de kaynaklanmaktadır. Karma-heterojen veri olması durumunda, verilerin birkaç modeli ve kuralları vardır ve modellerin özellikleri büyük ölçüde değişir. Veriler yapılandırılmış, yarı-yapılandırılmış ve yapılandırılmamış veriler olabilir. Kurumların ürettiği verilerin %80'i yapılandırılmamıştır. Bu veriler son derece dinamiktir ve belirli bir formatları yoktur. E-posta ekleri, resimler, pdf belgeleri, tıbbi kayıtlar, X ışınları, sesli postalar, grafikler, videolar, sesler vb. şeklinde olabilirler. Yapılandırılmış veriler ise satır/sütun formatında saklanır. Daha sonra analiz için bu verileri yapılandırılmış formata dönüştürmek, big data analizinde büyük bir sorundur. Dolayısıyla, bu tür verilerle uğraşmak için yeni teknolojiler benimsenmelidir [66].

2. Eksik veriler, veri analizi sırasında belirsizlikler oluşturulur ve veri analizi sırasında yönetilmelidir. Bunu doğru yapmak da bir zorluktur. Tamamlanmamış veriler, bazı örnekler için veri alanı değerlerinin kayıp olduğunu ifade eder. Eksik değerler, bir sensör düğümü arızası veya bazı değerleri kontrol altında atlayan sistematik arızalar gibi farklı durumlardan kaynaklanabilir. Modern veri madenciliği algoritmalarının çoğu, eksik değerleri işlemek için klasik çözümlere sahip olsa da (veri değerlerini eksik değerlerle göz ardı etmek gibi), veri yükleme geliştirilmiş modeller üretmek için eksik değerleri ortaya çıkartmaya çalışan yerleşik bir araştırma alanıdır. Bu amaç için çok sayıda yöntem vardır ve başlıca yaklaşımlar, en sık gözlenen değerleri doldurmak veya belirli bir örneğin gözlenen değerlerine dayanarak, her veri alanı için olası değerleri tahmin etmek için öğrenme modelleri oluşturmaktır [66].

3. Verinin depolanması konusunda, üretilen verilerin depolama alanları yetersizliği de big data'nın problemlerinden birisidir. Zira, her gün 2,5 ZB ortalama veri üretilmekte ve bu değer üç yılda bir iki katına çıkmaktadır [67]. Depolama alanlarının yetersiz kalması nedeniyle, uzaktan eğitim, sağlık sektörü gibi eski kayıtların çok önemli olmadığı durumlarda geçmiş veriler silinmektedir [68]. Silinen bu verilerin depolanabilmesi için ilave depo alma alanları kullanmak gerekmektedir. Bu da sisteme ve iş dünyasına yeni maliyetler getirmektedir. Aynı zamanda beklenen depolama alanları için sistem yönetim ve bakım maliyetleri ortaya çıkacaktır. Bu

sorunları ortadan kaldırmak için, dağıtık dosya sitemleri ve bulut sistemleri kullanılabilmektedir.

4. Scalability (ölçeklenebilirlik): Büyük ve hızla artan veri hacimlerinin yönetilmesi zorlu bir konudur. Geleneksel yazılım araçları, artan veri hacimlerini yönetmek için yeterli değildir. Veri analizi, yönetim, geri alma ve modelleme, analiz edilmesi gereken verilerin ölçeklenebilirliği ve karmaşıklığı nedeniyle de bir zorluk olarak ortaya çıkmaktadır. Veri hacmi küçük olduğunda geleneksel sistemler bir sorun ortaya çıkartmamaktadır ancak veri big data seviyesine ulaştığında sistem yapılan işlem isteklerine cevap verememekte ve problem ortaya çıkmaktadır. Veri büyüdükçe scalability (ölçeklenebilirlik) problemi ortaya çıkmaktadır.

Ortaya çıkan scalability probleminin etkisini azaltabilmek için yatayda ölçeklenebilen NoSQL veri tabanı sistemleri kullanılarak çözüm aranmıştır. Ayrıca Dağıtık Sistemler (Distributed Systems) ve bulut (cloud) teknolojisi de kullanılmaktadır [69].

5. Timeless: İşlenecek veri kümelerinin boyutu arttıkça, analiz ve yönetim daha fazla zaman alacaktır. Bazı durumlarda analiz sonuçları hemen gereklidir. Örneğin, sahte bir kredi kartı işleminden şüpheleniliyorsa, işlemin gerçekleşmesini engelleyerek işlem tamamlanmadan önce ideal olarak işaretlenmelidir. Açıktır ki, bir kullanıcının satın alma geçmişinin tam bir analizi gerçek zamanlı olarak mümkün olmayabilir. Bu yüzden kısmi sonuçları önceden geliştirilmesi gerekiyor, böylece yeni verilerle az miktarda ek bir hesaplama hızlı bir sonuca varmak için kullanılabilir.

Büyük bir veri kümesi verildiğinde, genellikle belirli bir ölçütle eşleşen öğeler bulmak gerekir. Veri analizi sırasında, bu tür aramaların tekrar tekrar oluşması muhtemeldir. Uygun öğeleri bulmak için tüm veri kümesinin taranması çok zaman alabilir. Bu gibi durumlarda, niteleyici öğeleri hemen bulmak için önceden indeks yapıları oluşturulmuştur.

Sorun, her indeks yapısının sadece bazı kriter sınıflarını destekleyecek şekilde tasarlanmış olmasıdır [66].

6. Kişisel Bilgi Güvenliği: Big data çağında internet servis sağlayıcılar, mobil şebeke operatörleri, büyük mağazalar, yerel kamu kurumları, tıbbi ve finansal hizmet kuruluşları (örneğin hastaneler, bankalar, sigorta ve kredi kartı kurumları) yoluyla kişisel bilgiler saklanmakta ve iletilmektedir. Ayrıca facebook, twitter gibi sosyal paylaşım sitelerinde de kişisel bilgiler paylaşılmakta ve saklanmaktadır. Gizlilik herkes için önemli bir konudur. Tüm bu bilgilerin yanlış kullanılmasını önlemek için kişisel bilgilerin gizlemesi istenmektedir. Fakat big data büyüdükçe, kişisel bilgilerin güvenliğinin sağlanması çok zorlaşmaktadır.

1.2.4. Big Data Çözüm Mimarileri

Elektronik belge yoluyla oluşturulan big datanın yönetim ve depolama problemini çözmek için dünya çapında, öncelikli olarak üç farklı çözüm yöntemi kullanılmıştır. Bunlar;

1. Veri Tabanı Sistemleri
   a. İlişkisel Veri Tabanı Sistemleri (RDBMS)
   b. NoSQL Veri Tabanı Sistemleri

2. İşletim Sistemi Dosyalama Sistemi

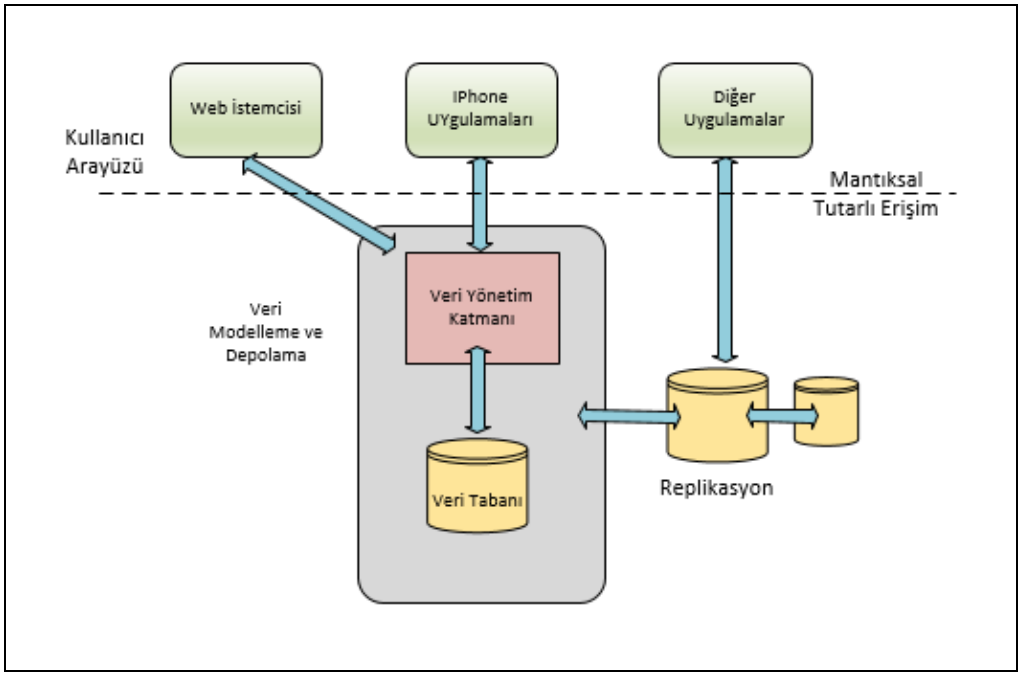
3. Dağıtık Dosya Sistemleri (DFS)

1.2.4.1. İlişkisel Veri Tabanı Sistemleri

Birincisi, iki farklı yaygın uygulama mimarisine sahip olan veri tabanı yönetim sistemidir; bunlar, ilişkisel veri tabanı yönetim sistemleri ve NoSQL veri tabanı

sistemleridir. Bu amaç için en yaygın bir şekilde bilinen ve kullanılan sistemler olan İlişkisel Veri Tabanı Yönetim Sistemleri, verilerin yapılandırılmış bir biçimde saklandığı depolama yöntemleridir. İşlenecek veriler uygun tip ve formatta olmalıdır. Bu sistemlerde, tek bir veri tabanı birden çok kullanıcıya ve uygulamaya hizmet edebilir. Bu sistemler dikey büyüme işlevselliği üzerine kurulduğundan, veri yapısı önceden tanımlanmalıdır ve ayrıca bu sistemler, atomicity, consistency, isolation, durability (ACID) şeklinde bir kısıtlamaya sahiptir. Bu sistemleri vazgeçilmez kılan bu katı kurallar bugün sorgulanmaya başlanmıştır. Ayrıca, kullanılan donanım ve yazılım nedeniyle, ilk kurulum maliyetleri yüksektir. Özellikle, veri hacmi arttığında, yatay ölçeklenebilirlik özelliği oldukça yetersiz ve yönetilmesi zor olmaktadır ki bu da big data sorunun çözümünün bir parçası olamamasının ana faktörü olarak ortaya çıkmaktadır. Ayrıca, bu sistemler, big data için uygun olmayan dosya sistemlerinden daha karmaşıktır.

1.2.4.2. NoSQL Veri Tabanı Sistemleri



Şekil 1.5. NoSQL Mimarisi

İlişkisel Veri Tabanı Yönetim Sistemlerinin big data'yı yönetme eksikliğinden dolayı, NoSQL veri tabanı sistemleri bir alternatif olarak ortaya çıkmıştır. NoSQL mimari yapısı Şekil 1.5'de gösterilmiştir [70]. Bu sistemlerin temel amacı, internetin artan veri hacimlerini depolayabilmek ve yüksek trafikli sistemlerin ihtiyaçlarına, yapılandırılmamış veya yarı yapılandırılmış formatlar üzerinden cevap verebilmektir.

NoSQL veri tabanı sistemleri 5 tipte kategorize edilmişlerdir. Bunlar;

- Anahtar-Değer Tabanlı Veri Tabanları (Key-Value Store Databases): Küçük ve çok sayıda okuma yazma işleminin yapıldığı uygulamalar için uygundur. Bir anahtara karşılık gelen veri genellikle boolean, integer gibi basit verilerdir. Bu tür veri tabanları, önbellek (caching) yazılımları, alışveriş sepeti uygulamaları ve görüntü dosyalarının saklanması gibi uygulamalar için uygundur. Anahtar değer veri tabanları, çevrimiçi alışveriş için web siteleri, forumlarda kullanılabilir. Amazon'un DinamoDB mimarisini tanıtmasıyla uzun zamandan beri var olan anahtar-değer veri tabanlı veri tabanlarının yanı sıra çok sayıda yeni anahtar değer veri tabanının geliştirilmesi için çalışmalar yapılmış ve ortaya çıkarılmıştır. Amazon DynamoDB, RIAK.

- Doküman Tabanlı Veri tabanları (Document Store Databases): Dokümanlar XML, PDF, JSON vb. gibi standart formatlardadır. İlişkisel Veri tabanlarında, aynı veri tabanındaki bir kayıt aynı veri alanlarına sahip olmak zorundadır ve kullanılmayan veri alanları boş tutulacaktır. Ancak doküman depolanacağında, her doküman benzer ve benzer olmayan veriye sahip olabilmektedir. Veri tabanındaki dokümanlar, bu dokümanı temsil eden benzersiz bir anahtar kullanılarak adreslenir. Bu anahtarlar, basit bir string veya URI veya path'e başvuru yapan bir string olabilir. Doküman tabanlı veri tabanları, anahtar-doküman çiftleri olarak da bilinen dokümanda anahtar-değer çiftlerini kaplamasına izin verdiği için anahtar-değer depolarına kıyasla biraz daha karmaşıktır. Doküman tabanlı veri tabanlarının en önemli özelliği

“esnek” olmalarıdır. Bir anahtara karşılık gelen veriler “doküman” adı verilen nesnelerde saklanırlar. Nesneler genellikle JSON formatındadır. Dokümanlar çok sayıda alan içerebilir ve her dokümanın yapısı birbirinden farklı olabilir. İlişkisel veri tabanlarında bu tarz çok biçimli (polymorphic) veriler çok sayıda tabloya dağılmış olarak saklandığı için karmaşık sorgular gerektirmektedir. Doküman tabanlı veri tabanları esnek yapısı ile bu ihtiyacı ortadan kaldırmaktadır. Doküman tabanlı veri tabanları, içerik yönetim sistemleri, elektronik ticaret uygulamaları ve günlük (blog) siteleri gibi esnek veri yapısına ihtiyaç duyan uygulamalar için uygundur. Doküman tabanlı veri tabanları, içerik yönetim sistemleri, elektronik ticaret uygulamaları ve günlük (blog) siteleri gibi esnek veri yapısına ihtiyaç duyan uygulamalar için uygundur. MongoDB, CouchDB.

- Graph Tabanlı Veri tabanları: Graph tabanlı veri tabanlarında veriler düğümler (node), ilişkiler (edge) ve özellikler (properties) şeklinde tutulurlar. Diğer veri tabanı türlerinden farklı olarak veriler arasındaki ilişkiler de saklanabilir. Diğer NoSQL veri tabanı türleri çok geniş kullanım alanına sahipken, çizge tabanlı veri tabanlarının kullanım alanı daha kısıtlıdır. Çizge tabanlı veri tabanları, BPM uygulamaları, sosyal ağ uygulamaları, kimlik ve erişim yönetimi uygulamaları ve tavsiye motorları gibi uygulamalar için uygundur. Grafik veri tabanlarında “sharding“ elde etmek çok zordur. Grafik veri tabanlarında cluster yapılandırması oldukça zordur. Neo4j, grafik veri depolarını kullanan dikkate değer DBaaS sağlayıcılarından biridir.

- Sütun Tabanlı Veri tabanları (Column Store Data Bases): Sütun tabanlı veri tabanları, yüksek okuma yazma performansı ve yüksek erişilebilirlik (high availibity) için tasarlanmıştır. Birden çok sunucu üzerinde dağıtık olarak çalışırlar ve bu sayede tek bir sunucuda tutulamayacak kadar büyük verileri saklayabilirler. Yazma işleminde kesinti yaşanmaz fakat dağıtık yapısından dolayı kısa süreli veri tutarsızlığı (inconsistency) yaşanabilir. Bu özelliği tolore edemeyen uygulamalar için uygun değildir. Kolon tabanlı veri tabanları, içerik yönetim sistemleri, günlük (blog) uygulamaları, uygulama

kayıtlarının (log) saklanması gibi uygulamalarda yaygın olarak kullanılmaktadır. Column tabanlı veri tabanları, veri madenciliği ve analitik uygulamalar için uygundur, burada depolama yöntemi, veriler üzerinde gerçekleştirilen ortak işlemler için idealdir. Column tabanlı veri tabanları kullanan dikkate değer DBaaS sağlayıcılarından bazıları şunlardır: Big Table, Cassandra.

- Nesne Tabanlı Veri Tabanları (Object Oriented Databases): Nesneye yönelik bir veri tabanı, saklanacak verilerin veya bilginin bir nesne olarak temsil edildiği bir veri tabanıdır (nesne yönelimli programlama dili kavramında kullanılan bir nesneye benzer). Böylece nesne yönelimli veri tabanı, nesne yönelimli programlama (OOP) ve veri tabanı prensiplerinin bir kombinasyonu olarak düşünülebilir. Nesne veri deposu, veri kapsülleme, polimorfizm ve kalıtım gibi OOP'un tüm özelliklerini sunar. Bu tür veri tabanlarındaki sınıf, nesne ve sınıf özellikleri, sırasıyla RDBMS'deki bir tablo, satır ve sütunlarla eşleştirilebilir. Her bir nesne, bu nesneyi benzersiz bir şekilde temsil etmek için kullanılabilecek bir nesne tanımlayıcısına sahiptir. Nesne yönelimli veri tabanları kullanıldığında verilere erişim daha hızlıdır, çünkü nesne doğrudan işaretçiler kullanılarak alınabilir. Nesne yönelimli veri tabanları, modern yazılım geliştirme süreçlerinin çevik olmasını kolaylaştırır. Db4o, Nesne yönelimli veri tabanlarını kullanan bir DBaaS sağlayıcısıdır [71, 72]

NoSQL veri tabanları, İlişkisel Veri Tabanı Yönetim Sistemine göre yüksek erişilebilirlik sağlayan ve verilerin yatay olarak kolayca ölçeklendirilebildiği sistemlerdir [73]. Okuma ve yazma performansları İlişkisel Veri Tabanı Yönetim Sistemlerine göre daha kabul edilebilir olabilmektedir. En önemli özelliklerden biri, yatay olarak genişletilebilmesidir. Binlerce sunucu bir küme olarak birlikte büyük veriler üzerinde çalışabilir. Esnek yapıları sayesinde programlamak ve yönetmek kolaydır. Bu sistemlerin bir diğer özelliği, ağa bağlı birçok makineden oluşan cluster-kümelerde grid hesaplama yapabiliyor olmalarıdır, bu şekilde veri işlem hızları artmaktadır. Bunların yanında, veri güvenliği konusunda henüz İlişkisel Veri Tabanı

Yönetim Sistemleri kadar gelişmiş özelliklere sahip değildir. Bazı NoSQL projeleri dokümantasyon konusunda ve profesyonel teknik destek konusunda hala eksikliğe sahiptir. Ayrıca, veri kaybına neden olabilecek transaction kavramı NoSQL veri tabanı sistemlerinde mevcut değildir, bu nedenle bankacılık ve finansal sistemlerde kullanım için uygun değildir [74].

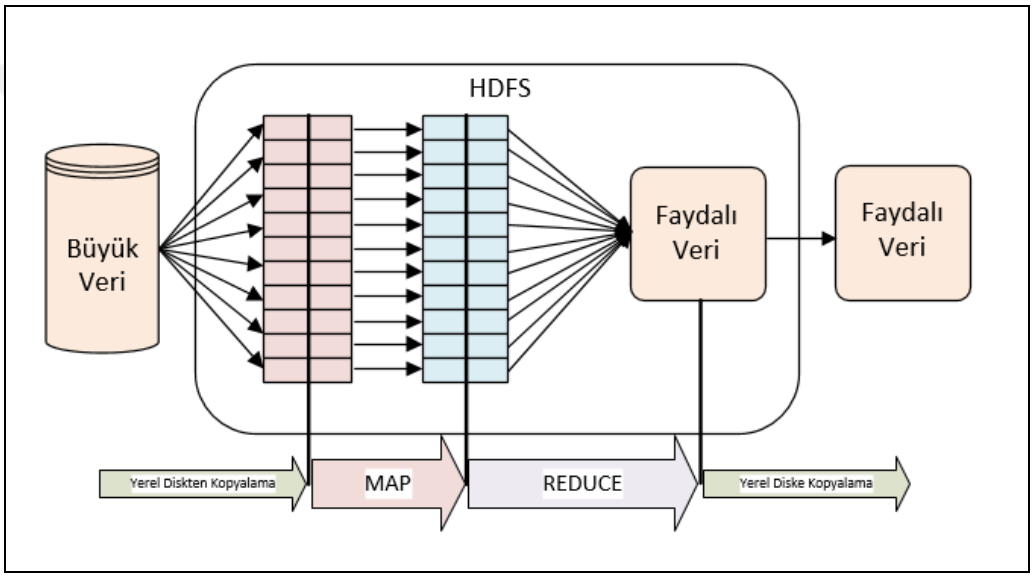
1.2.4.3. İşletim Sistemi Dosyalama Sistemi

Literatürde "file server" olarak adlandırılan işletim sistemlerinin temel dosya yönetim fonksiyonu ikinci çözüm yöntemidir. Bu sistemlerde, EBYS'de üretilen veriler, temel işletim sistemi dosya yapısındaki dosya ve klasörlerde saklanır. İşletim sistemleri hiyerarşik bir dosya sistemi kullanır. Bu yapıda, dosyalar bir "dizin" olarak adlandırılan bir ağaç yapısındadır. File serverlar, dosyaları dosya adı, dosya türü, oluşturulma tarihi, ait olduğu klasör adına göre, EBYS tarafından tanımlandığı şekilde depolar. EBYS, işletim sistemi yardımıyla depolama cihazlarına düşük seviyeli ara yüz sağlayan sistem çağrılarını çağırarak okuma ve yazma işlemlerini gerçekleştirir. File serverları kullanmanın avantajları, bunların dağıtımının basit, uygulanmasının kolay ve yeterli sayılabilecek dosya işlemi performansına sahip olmalarıdır. İşletim sistemi üzerinde dosya yazmak, silmek, okumak ve aramak oldukça hızlı bir süreçtir. Özellikle, İlişkisel Veri Tabanı Yönetim Sistemleri ve NoSQL sistemlere göre işletim sistemleri daha fazla esnekliğe ve performans özelliğine sahiptir. Ancak bununla birlikte, bu avantajlarını yatay ölçeklenebilirlik eksikliği nedeniyle big data için yeterli bir çözüm modeli olarak kullanamaz İşletim sitemleri dosya yönetiminin ana görevi, diğer uygulamalara sistem çağrısı hizmeti vermektir. Bu yüzden, tek başına bir çözüm değil, çözümün bir parçası olarak kullanılmaktadır. Bunun yanında, diğer yöntemler kadar veri depolamada güvenli olsa da veri boyutuna göre ölçeklendirme ve yedekleme yapamaz ve bu sistemde veri transferini güvenli bir şekilde yapılamaz. İşletim sisteminin big data problemlerini çözmek için tek başına uygun olmadığı görülmektedir.

1.2.4.4. Dağıtık Dosya Sistemleri

Üçüncü yöntem, Distributed File System (Dağıtık Dosya Sistemi (DFS)'dir. Bu sistemler, farklı konumlardaki makinelerin tek bir framework olarak çalıştığı ve big data sorununa en uygun çözümü sağlayan en güncel sistemlerdir. En sık kullanılan dağıtık dosya sistemi Hadoop'tur.

Hadoop:



Şekil 1.6. Hadoop genel mimarisi [75]

Dağıtık sistemde birden fazla makinenin işlemcileri bir ağ üzerinden haberleşirler. Her işlemcinin kendisinin lokal belleği ve işlemcisi vardır. Bunlar dağıtık sistemde paylaşılmaz kendi görevlerini yerine getirmek için kullanılırlar [76]. Sistem kullanıcı tarafından tek ve bütünleşik bir yapı ve işlem gibi görünür. Donanım ve yazılım farklılıklarını ortadan kaldırır. Kullanıcı için dağıtık sistemde bir ara yüz var gibi düşünülse de işlem farklı bilgisayarlarda eş zamanlı olarak yapılmaktadır. Sunuculu sistemler ve peer-to-peer yapılar dağıtık sisteme örnek olarak gösterilebilir. Dağıtık

sistemlerde veriyi işleme sadece bir makine ile kısıtlanmamış, birden fazla bilgisayar üzerine dağıtılmıştır. Büyük işletme ve kurumların big data işlemleri bu şekilde dağıtık sistemler üzerinde yapılmaktadır. Dağıtık sistemlerde donanım ve yazılım kaynakları paylaşılmaktadır. Birden fazla makinenin eş zamanlı işlem yapabiliyor olmasıyla sistem performansı arttırılmış olmaktadır. Yazılım ve donanım uyumsuzluğu gibi bir problem olmamaktadır. Farklı donanım ve yazılım özelliklerine sahip makineler beraber sistemde çalışabilmektedir. Bütün bu olumlu özelliklerinin yanı sıra dağıtık sistemlerde merkezi bir yönetim ile sistem çalıştığından karmaşıklık vardır, yönetim zordur. Sistem yönetimi için ayrı bir personel istihdamı gerekmektedir [77].

Hadoop iki yapıdan oluşmaktadır: MapReduce ve Hadoop Distributed File System (HDFS). Şekil 1.6'da genel mimarisi gösterilmiştir. MapReduce, Google tarafından geliştirilmiştir ve önemli bir frameworktur. Dosyayı 64 MB'lık blok büyüklüğünde işleme tabi tutar. MapReduce üç işlem bloğuna ayrılabilir. Bunlar, Map, Reduce ve Ana bloktur. Map ana node işlemlerini yapar, veri girişi işlemlerini yapar. Gelen veriyi işlem yapabileceği bloklara böler ve data node'lara gönderir. Data node'lar işlemleri yaptıktan sonra tekrar Map işleminin yapıldığı name node'a gönderir. Reduce işleminde ise name node'a gelen veriler birleştirilir ve sonuç işlemi yapılır. Pek çok büyük işyerlerinde ve kurumlarda Hadoop kullanılmaktadır [77]. Hadoop birçok özelliği bakımından verilerin güvenli, etkin ve ölçeklenebilir olarak işlenmesini sağlar. Hesaplanan elemanların ve depolamanın başarısız olabileceği durumunu ortadan kaldırmak için data bloklarının replikasyonları alınır ve node'larda tutulur. Eğer bir node'un çalışmasıyla ilgili problem ortaya çıkarsa o node'ta bulunan data diğer replicadan alınarak işlem devam ettirilir. Bu şekilde Hadoop paralel işlem yapar. Bu sayede ölçeklemeyi de yapmış olur. Hadoop mimarisinin en alt katmanında HDFS (Hadoop Distributed File System) dosya sistemi bulunur. Dışarıdaki bir client HDFS'yi klasik bir dosya sistemi gibi görür. HDFS verileri düğümler şeklinde tutmaktadır. Dosya oluşturma, güncelleme, okuma ve silme işlemleri (CRUD)'ni yapar. Name node ve Data node'tan oluşmaktadır. Name node dağıtık dosya sisteminde metadata hizmeti verir, Data node ise dosyalama sisteminde veriyi depolama için bloklar sunar. HDFS içerisinde tüm

iletişimler TCP/IP protokolü ile yapılmaktadır. HDFS dosyaları 64 MB büyüklüğündeki bloklara böler. Bu blokları veri güvenliği açısından 3 kopya şeklinde replication yapar. Bu replikaları node'larda tutar. Replika sayısı kullanıcı tarafından güncellenebilmektedir [78]. Hadoop, Hive, Hbase, Pig, Mahout, Hama, Zookeeper gibi bileşenler barındırır. HBase, Hadoop'un çok büyük miktarda veriyi tutabilen veri tabanıdır. Milyonlarca satırda veri tutan bir table gibidir. Google BigTable'in açık kaynak karşılığıdır [77, 79].

Dağıtık Dosya Sistemi ve MapReduce temel olarak büyük veri depolama ve analiz için kullanılır. Hadoop ve NoSQL'in birlikte kullanıldığı hybrid çözümler de literatürde değerlendirilmiş ve kullanılmıştır. Bununla birlikte, elektronik belge yönetim sistemlerinde Hadoop ekosistemini kullanmanın bazı dezavantajları vardır. Birincisi, Hadoop 'un bu tür dosyaları verimli bir şekilde depolayamayacağı ve yönetemeyeceği anlamına gelen küçük dosyalar problemidir. Hadoop, esas olarak 64 MB'a eşit veya daha büyük dosyaları yönetmek için tasarlanmıştır. Bu boyut ayrıca Hadoop clusterlarının (kümelerinin) varsayılan blok boyutudur [80]. Örneğin, 64 MB'lık Hadoop bloklarından oluşan 16 MB'tan meydana gelen 1 GB'lık bir dosya, bir name node 'ta 2,4 GB yer kaplar. Ancak bununla beraber, 100'lık 100.000 dosya bir data node 'ta 1 GB'lık, name node 'ta 1,5 GB yer kaplar. Bu, küçük dosyaları işlerken gereğinden fazla MapReduce işlemi yapılacak anlamına gelir. EBYS'lerin ürettiği ve işlediği dosyalarda ortalama boyut 64 MB'tan küçüktür [47]. Bu durum göz önüne alındığında, çok sayıda küçük dosyaya sahip sistemleri yönetebilmek için yeni bir Dağıtık Dosya Sistemi sistemi tasarımına ihtiyaç duyulmaktadır. Bu çalışma, bu konu için küçük blok büyüklüğü ve name node kullanmadan yeni bir metodoloji önermektedir. Raghupathi ve Raghupathi tarafından yürütülen bir çalışmada, Hadoop ekosisteminin orta veya büyük ölçekli sektörlere hizmet sağlarken önemli dezavantajlara sahip olduğunu açıklanmıştır: (a) MapReduce kullanarak big data analizi işlemleri için çok fazla programlama becerisi gerektiriyor ve (b) Hadoop ekosistemini tamamen kurmak, yapılandırmak, yapılandırmak çok zordur, bu yüzden orta veya büyük ölçekli sağlık hizmeti sağlayıcıları için uygun bir çözüm gibi görünmemektedir [81].

1.3. Elektronik Belge Yönetim Sistemi

1.3.1. Elektronik Belge Yönetim Sistemi Kavramları

1.3.1.1. Veri (Data)

Üzerinde işlem yapılmamış, sensörlerden ve aygıtlardan gözlemle veya belirli ölçümlerle elde edilmiş değeri olan her şeydir [82]. Başka bir kaynakta veriyi, tek başına olduğunda bir anlamı olmayan, kullanılamayan, ancak bilgi ve bilgi yönetimine temel oluşturan ilişkilendirilmeye, gruplandırılmaya, yorumlanmaya, anlamlandırılmaya ve analiz edilmeye gereksinim duyulan ham bilgi şeklinde tanımlamıştır [83].

Veriler aşağıda verildiği gibi gruplara ayrılarak nitelendirilebilir [84]:

- Yapılandırılmış, yapılandırılmamış, yarı yapılandırılmış
- Statik, dinamik, akan
- Güvenli / açık, özel / halka açık
- Ücretli / ücretsiz
- Açık kamusal veri
- Açık veri
- Big data

Big data konusunun temelini, verinin bu kadar değişik şekillerde gruplandırılması ve önceden bu grupların çok azının yönetilebiliyor olması oluşturmaktadır [85].

1.3.1.2. Bilgi (Information)

Veri hamdır, işlenmemiştir, 1 ve 0'lardan oluşan, insanlar için anlamsız bir değerdir. Veri işlendikten sonra anlam kazandığında bilgi halini almış olur. Veri işlendikten sonra bilgiye dönüşür. İnsan dostu hale gelir. Bir yapı ve anlam kazanmış olur. Sonuca ulaşmak için karar vermede faydalı olur. Veriler bir araya geldiklerinde ve işlendiklerinde bilgiyi oluşturmuş olurlar.

Veri ile bilgi arasındaki farkları özetlemek gerekirse:

- Veri bilgisayar sistemleri için bir girdidir buna karşılık bilgi, girdinin bir çıktısıdır.
- Veri işlenmemiş rakamlar, sayılar ve şekillerdir. Bilgi verilerin işlenmiş halidir.
- Veri bilgiden bağımsızdır. Bilgi ise veriye bağımlıdır.
- Veri özel, kesin, tanımlı değildir. Bilgi özeldir, belirlidir.
- Veri tek bir parçadan oluşur. Bilgi ise gruplandırılmış verilerden oluşabilir.
- Veri anlamsızdır. Bilginin bir anlamı vardır.
- Veri hamdır. Bilgi üründür [86].

1.3.1.3. Doküman

Belge ile doküman kavramı arasındaki temel farkları açıklamak gerekmektedir. Doküman, resmi bir nitelik taşımayan fakat kurumsal faaliyetlerin yürütülmesinde bilgi amaçlı başvurulan kayıtlı bilgi ve nesnelerdir [87]. Doküman, baskı yoluyla kaydedilmiş metin, ses, resim, hareketli resim gibi bilgilerden oluşmaktadır; fakat hem idari hem de hukuki olarak delil teşkil edemez.

1.3.1.4. Belge

“Resmi Yazışmalarda Uygulanacak Usul ve Esaslar Hakkında Yönetmelik’te belge, herhangi bir bireysel işlemin, kurumsal fonksiyonun veya kurumsal işlemin yerine getirilmesi için alınmış ya da idare tarafından hazırlanmış; içerik, ilişki ve formatı ile ait olduğu fonksiyon veya işlem için delil teşkil ederek aidiyet zincirini muhafaza eden, el yazısı ya da güvenli elektronik imza ile imzalanmış ve EBYS ya da kurumsal belge kayıt sistemleri içinde kayıt altına alınmış her türlü kayıtlı bilgi veya doküman olarak tanımlanmıştır. Yine aynı Yönetmelikte doküman, kurumsal faaliyetlerin yerine getirilmesi amacıyla idare tarafından hazırlanan ya da toplanan her türlü bilgi olarak ifade edilmiştir [88].

1.3.1.5. Elektronik Belge (Electronic Document)

Model Requirements for the Management of Electronic Records (Elektronik Belge Yönetimi Model Gereksinimi) Moreq’e göre elektronik belge; “Bir yazılım tarafından elektronik ortamda üretilen ya da kâğıt veya mikro formdan dönüştürülerek dijitalleştirilen elektronik kayıtlardır.” Karakaş ve arkadaşlarının yazdığı Belge Yönetimi ve Arşiv Terimleri Sözlüğü ’ne göre elektronik belge; “Çeşitli bilgisayar ve taşınabilir elektronik araçlar gibi bilgi ve iletişim teknolojileriyle oluşturulan, herhangi bir kurum ve/veya kişiden başka bir kurum ve/veya kişiye aktarılan veya aynı ortamlarda da saklanan her türlü yazılı, sesli ve görüntülü kaydedilmiş bilgi şeklindedir.”. B. K. Ataman Tarafından yazılan Arşivcilik Terimleri Sözlüğünde elektronik belge, "Genellikle kodlanmış manyetik disk, manyetik bant, delikli kart gibi ortamlar üzerine kayıtlı, içerikleri yalnızca makineler aracılığıyla okunabilen ve veri arşivlerinden farklı olarak provenans ilkesine uygun biçimde düzenlenmiş evraklar/arşivler" şeklinde tanımlanmıştır [89].

1.3.1.6. Evrak

Türk Dil Kurumuna göre evrak; resmî kurumlarda işlem gören Kâğıt yaprakları, kitap sayfaları, yazılmış kitaplar, mektuplar veya yazıların tümü olarak açıklanmıştır.

1.3.2. Belge Yönetimi

Kurum ve kuruluşta üretilen bilginin bir değer olarak tutulabilmesi için, onun sürekli olması ve kendine ait özelliklerinin olması gerekir. Bilgi bir faaliyetin sonucu olarak ortaya çıkan ve yönetim süreci sonunda ortaya çıkmış planlı düzenli bir üründür. Bilişim Teknolojileri (BT), bu sürecin en önemli araçlarıdır. Bilgi yönetiminin başarılı olabilmesi için hem belge yönetiminin hem de BT'nin etkin bir şekilde kullanılması gerekmektedir. Ayrıca üretilen bilginin bazı ortak platformlarda da yayılması ve dağıtılması gerekebilmektedir [90].

Belge yönetiminin amacı, ilk önce belgenin güvenliğinin sağlanmasıdır. Belge oluşturulduktan sonra, belgesini takibi ve aramasının da sağlıklı yapılabilmesi belge yönetiminin bir konusudur. Belgeyi depolamak ve gerektiği zaman kullanıcıya geri döndürmek belge yönetiminin unsurudur. Belge yönetimi, önemli/önemsiz belge ayırımını yapar, gerekli imha işlemlerini yapar, depolama maliyetini düşürür, verimliliği artırır. Bu sayede bilgiye daha kolay erişilmiş olur [91].

ISO 15489, Uluslararası Belge Yönetim Standardında belge yönetimin tanımı aşağıdaki şekilde yapılmıştır: Bir işin yapılması sürecinde, işlemler ve süreç ile ilgili bilgileri saklayan, belgelerin etkili ve düzenli bir şekilde kontrol edilerek belgenin üretilmesi, korunması, alınması, iletilmesi, imzalanması, imha edilmesi ve istenildiği zaman bulunması ile ilgili süreci yöneten çalışma alanıdır [87].

1.3.3. Elektronik Belge Yönetim Sistemi (EBYS):

Oluşturulan belgelerin etkin bir şekilde yönetilebilmesi için bir belge yönetimine ihtiyaç vardır. Belgelerin sadece elektronik ortamda tutuluyor olması onların yönetilebilir bir ortamda bulunduruluyor ve gerektiği zaman kullanılabilir bir şekilde depolandığını göstermez. Depolama işlemi dijital ortamda yapılacaksa o zaman e-belge yönetim sistemi kullanılmalıdır. Dahası e-belgelerin oluşturulmadan önce EBYS'nin kullanılmaya başlanması gerekmektedir. EBYS tasarımında belge yönetim sürecinin aşamaları göz önünde bulundurulmalıdır. Sistem tasarımının teorik ilkeleri gerçekleştirilmelidir. Ayrıca, bir belgenin nasıl üretileceği, nasıl tanımlanacağı ile ilgili tanımların yapılarak sistematik bir şekilde saklama sureleri ve imha tarihleri tespit edilmelidir. Güvenli erişim sağlanabilmeli ve verilerin kişiler tarafından hangi yetkilerle erişilebileceği, erişim kısıdının sağlanması şifreleme, imzalama yetkisi, iş akışı gibi durumların ve koruma, belli belgelere erişimin kısıtlanması ve kimin yetkili olduğunu belirleyen kararının alınması gerekmektedir [92, 93]. Bir kurumda Elektronik Belge Yönetim Sistemi kullanıldığında şu faydaları sağlayacağı düşünülmektedir; hız, güvenlik, maliyet, verimlilik, bilgi yönetimine katkı.

1.3.4. Elektronik Belge Yönetim Sistemi İle İlgili Çalışmalar:

Elektronik Belge Yönetim Sistemi çalışmaları iki başlık altında incelenmiştir.

1.3.4.1. Dünyada EBYS Standardı İle İlgili Çalışmalar:

Uluslararası Standartlar Örgütü (International Organization for Standardization-lSO) 1996 yılında Avustralya'da yayımlanan, AS 4390 Avustralya Belge Yönetim Standardı (Australian National Standard for Records Management)'nı ilk belge yönetim standardı olarak kabul etmiştir. Belge yönetim sisteminde ulusal ilk standart kabul edilen AS 4390, temel alınarak Uluslararası Standartlar Örgütü tarafından

uluslararası nitelikteki ilk belge yönetimi standardı olan "ISO 15489 Enformasyon ve Dokümantasyon- Belge Yönetimi (Information and Documentation- Records Management)" olarak geliştirilmiştir [94].

Avrupa Birliğinde ise, ilk standartlaşma çalışması 2001 yılında yapılmıştır. İngiltere'de İngiliz Ulusal Arşivi (British Public Records Office) ve Avrupa Birliği (AB) tarafından "Elektronik Belge Yönetimi Model Gereksinimi (Model Requirements for Electronic Records Management- MoReq) geliştirilen EBYS standardıdır. Bu standart, 1997 yılında Amerikan Ulusal Arşivi (National Archives and Records Administration)'nin katkılarıyla yayınlanan DoD 5015.2 "Elektronik Belge Yönetim Yazılım Uygulamaları İçin Standart"'ını temel almıştır [95].

1.3.4.2. Türkiye' de EBYS İle İlgili Çalışmalar:

Ülkemizde de elektronik belgelerin kayıt altına alınması, kullanılması ve arşivlenmesi konularında çalışma yapma görevi Devlet Arşivleri Genel Müdürlüğü ne verilmiştir [96].

Dünyadaki ulusal belge yönetimi standartlarına bakıldığında, ulusal olarak gördüğümüz standart ölçütleri yerele uyarlanarak güncellenmiş uluslararası ölçütlerdir. Türk Standartları Enstitüsü (TSE) tarafından da kabul edilen ve hemen hemen tüm ülkelerde de sürecin bu şekilde işlediği görülen Türkiye Ulusal Belge Standardı da ISO'nun kabul ettiği standardın içeriği ile yaklaşık aynıdır [97]. Bu çerçevede TSE'ye bağlı bir birim olan Bilgi Teknolojileri ve İletişim İhtisas Grubu, ISO 15489 Uluslararası Belge Yönetimi Standardını ve teknik raporunu Türkçeye çevirmiştir. Bunu Temmuz 2007'de "Bilgi ve Dokümantasyon- Belge Yönetimi (TSE ISO 15489)" adı ile standart olarak kabul edilmiştir [98].

1.3.5. Elektronik Belgelemede Big Data Problemi

Elektronik Belge Yönetimi ve arşivleme alanında, big data problemleri ve önceki argümanlara göre çalışmanın amaçları aşağıdaki gibi sıralanmıştır:

1. Artan belge sayısı: Dünya nüfusunun artması, iş ve ticaretin artması ve teknolojinin de kullanımının artması üretilen belge sayısını arttırmıştır. Örnek kuruluşların evrak artışları Çizelge 1.2, Çizelge 1.3, Çizelge 1.4 ve Çizelge 1.5'te gösterilmiştir.

Çizelge 1.2. Yıllar itibarıyla örnek kurumda evrak sayısı (Yalova ÇŞ İl Müd.) [99]

Yılı	Gelen Evrak	Giden Evrak	Genel Toplam
2010	2618	1997	4615
2011	4230	3166	7396
2012	8826	6196	15022
2013	8480	5666	14146
2014	8594	5095	13689
2015	7202	4213	11415
2016	7014	4014	11028
2017	6042	4351	10393

Çizelge 1.3. Yıllar itibarıyla örnek kurumda evrak sayısı (İzmir EMO) [99]

Yılı	Gelen Evrak	Giden Evrak	Genel Toplam
2010	3318	3800	7118
2011	3549	3997	7546

2. Teknolojik cihazlar: Daha kapsamlı dijital belge üreten cihazların artması, depolanacak büyük miktarda veri anlamına gelen yüksek çözünürlüklü grafik çıktılar ve dosyalar oluşturur.

Çizelge 1.4. Yıllar itibarıyla örnek kurumda evrak sayısı (KTÜ-İİBF) [99]

Yılı	Kurum Dışı Gelen	Kurum Dışı Giden	Kurum İçi Yazışma
2014	42	5	1975
2015	43	81	2625
2016	27	246	3314
2017	39	211	3241

Çizelge 1.5. Kırıkkale Üniversitesi, EBYS veri büyüklüğü ve dosya adedi

Yıl	Veri Büyüklüğü	Dosya Adedi
2015	341 GB	1.500.000
2016	673 GB	1.700.000
2017	550 GB	1.600.000
2018 ilk yarısı	370 GB	870.000
TOPLAM	2,37 TB	5.670.000

3. Uzman personel ihtiyaçları: Hadoop, Spark, Kubernetes, Elasticsearch vb. kullanarak kurumlardaki büyük verileri yönetmek, büyük bilgi çözümlerini dağıtmak, veriyi saklamak için nitelikli bilgi teknolojisi uzmanları kullanılmalıdır

4. Küçük dosya boyutu sorunu: Hadoop tabanlı çözümlerin de dahil olduğu elektronik belge yönetim sistemleri için çözümler, bir sonraki bölümde ayrıntılı olarak anlatılacak olan 64 MB'lık bir blok boyutuna sahiptir; Bu,

performansta ve "dahili parçalanma" olarak adlandırılan gereksiz depolama alanı kullanımındaki güvenlik açıklarına yol açar ve bu da çözülmesi zordur.

5. Elektronik Belge Yönetim Sistemleri (EBYS): Bu sistemler, elektronik belge hizmeti sağlayıcılarının elektronik belge bilgilerini daha verimli bir şekilde üretmelerini, saklamasını, veri almasını ve verileri değiştirmesine yardımcı olan kapsamlı yazılım ve ilgili araçlardır ve daha iyi belge takibi ve sistem bakımı sağlar. EBYS, (a) sağlamlık, (b) performans, (c) ölçeklenebilirlik ve (d) kullanılabilirlik gibi temel işlevsel özelliklere sahip olmalıdır. Bu özellikler temel olarak yapılandırılmış veri yönetim mimarisine dayanır.

1.4. Temel Teknolojiler ve Algoritmalar

Tasarlanan dağıtık dosya sisteminde verinin depolanması ve yönetilmesi için teknolojiler kullanılmıştır.

1.4.1. TCP/IP Protokolü

Protokoller, yerel alan ağları veya geniş alan ağlarında bulunan iletişim nesneleri arasında mesaj alışverişi için kullanılan kesin kurallar dizisidir. İnternet'teki her bilgisayar veya mobil cihaza, İnternet'teki diğer cihazlarla çakışmayan eşsiz bir İnternet Protokolü (IP) adresi verilir. Internet Protokolü, bir IP adresi ve bir port numarasının kombinasyonudur. Gönderici ve alıcı aygıtların IP adreslerini ve ilgili veriyi içeren datagramları veya paketleri içerir. Sonuçta, endpointlere bağlanmak için IP adreslerini kullanır. Bu IP, User Datagram Protocol (UDP), Lightweight User Datagram Protocol (UDP-Lite), Transmission Control Protocol (TCP), The Datagram Congestion Control Protocol (DCCP), Stream Control Transport Protocol (SCTP) gibi taşıma katmanı protokollerine ihtiyaç duyar. TCP ve UDP çoğunlukla internet bağlantılarında kullanılmaktadır. TCP'nin UDP'ye göre birçok avantajı

vardır: a) Güvenilirlik b) Paketlerin sipariş edilmesi c) Akış. Bu nedenle bu projede TCP / IP kullanılmıştır.

1.4.2. Socketler

Bir TCP soketi, bir IP adresi ve istemci/sunucu etkileşimi için bir port numarası tarafından tanımlanan bir işlemin uç noktasıdır. Bu nedenle, TCP socketi bir bağlantı değildir, iki süreç arasındaki iki yönlü bir iletişim bağlantısının sadece bir uç noktasıdır. İki tip socket vardır: her ikisi de veri gönderilip alabilen istemci ve sunucu socketleri. Dolayısıyla, aralarındaki ayrım, bağlantının nasıl oluşturulduğuyla ilgilidir. Sunucu socketi istemci istekleri için belirtilen bağlantı noktasını sürekli olarak dinlerken istemci socketleri, bağlantıyı başlatır. Bu projede .NET Framework Socket kütüphanesi kullanılmıştır.

1.4.3. Windows Servisleri

Windows Servisleri, Microsoft Windows işletim sisteminin özel bir tür süreçleridir. Servisler ve uygulamalar arasındaki farklar a) arka planda çalışırlar b) genellikle bir kullanıcı ara yüzüne veya kullanıcı ile etkileşime sahip değildir c) bilgisayar kapanana kadar uzun süren işlemlerdir d) bilgisayar yeniden başlatıldığında otomatik olarak başlarlar. Bu projede, Windows servis yordamları hem istemci hem de sunucu tarafı (Dağıtık Dosya Sistemi) uygulanmıştır. Bu servisler ayrıca, verileri karşılıklı olarak iletmek için istemci ve sunucu socket yapılarına sahiptir.

1.4.4. Şifreleme

Şifreleme, verilerin gizli tutulmasına izin veren, güvenli bir şekilde veri iletmek için ileri matematiksel teknikleri içeren bir yöntem olarak tanımlanır. Veriler, iletim başlamadan önce şifreleme anahtarı kullanılarak değiştirilir ve alıcı, bu değiştirilmiş

verileri uygun şifre çözme anahtarıyla orijinal verilere dönüştürür. Simetrik ve asimetrik olan iki tip şifreleme yöntemi yıllardır başarıyla kullanılmaktadır. Asimetrik şifreleme, public key metnini şifrelemek için iki anahtar (genel ve özel) kullanan ortak anahtar şifrelemesi olarak da bilinen daha yeni bir yöntemdir. Ortak anahtar şifrelemek için kullanılırken, özel anahtar şifreyi çözmek için kullanılır. Bu iki anahtar özel matematiksel tekniklerle oluşturulur. Genel anahtar herkes tarafından bilinebilir, ancak özel anahtar sunucu tarafında olmalı ve hiç kimse bunu bilmemelidir. Anahtarlar farklı olsa da matematiksel olarak birbirleriyle ilişkilidir. Özel anahtar gizli tutulur ve ortak anahtar, veri aktarımının gerçekleştirileceği hedef cihazlar ile kolayca paylaşılabilir; çünkü bu anahtarın bilinmesi, şifrelenmiş verilerin çözümüne katkıda bulunmayacaktır. Rivest-Shamir-Adleman (RSA), Digital Signature Algorithm (DSA), Diffie-Hellman, Pretty Good Privacy (PGP) yaygın olarak kullanılan asimetrik anahtar algoritmalarıdır.

Tek anahtarlı kriptografi olarak da adlandırılan simetrik kriptografi, hızlı, uygulanması en basit ve ilgili verileri şifrelemek ve deşifre etmek için sadece bir gizli anahtar içeren en iyi bilinen tekniktir. Simetrik anahtar algoritmaları öncelikli olarak blok veya veri akışlarının toplu şifrelenmesi için kullanılır. Data Encryption Standard, Advanced Encryption Standard, Stream Cipher Algorithm ve Blowfish gibi birçok Simetrik anahtar algoritması vardır. Blowfish, kullanımda en hızlı blok şifrelerden biridir ve 32 bit'ten 448 bit'e kadar anahtar uzunluğa sahiptir. Bu çalışmada 128 bit anahtar uzunluğu olan Blowfish algoritması tercih edilmiştir.

1.4.5. Dosya İşleme Yordamları

Bir işletim sisteminin temel görevi sağlam, hızlı ve verimli bir dosya sistemine sahip olmaktır. Tüm işletim sistemleri, fiziksel depolamayı blok diye isimlendirilen 4 KB 'lık alanlara eşit olarak böler ve bunları en verimli şekilde yönetmeye çalışır. Temel olarak iki farklı disk alanı yönetimi kavramı kullanılmaktadır. Birincisi Unix tabanlı bitmap yöntemidir. Bu yöntemde, blok depolama aygıtı kümelere bölünmüştür ve her bir kümenin karşılık gelen bir bit'i vardır, burada 1'ler boş disk bloklarını, 0'lar tahsis

edilen blokları temsil eder. Blokların sayısı kadar bit vardır. Bu map yapısı, dosya sistemine eklenecek dosyalar için erişilebilir boş disk bloklarını bulmak için kullanılır. Disk alanı ayırma için başka bir yöntem de Windows tabanlı işletim sistemlerinin başarıyla kullandığı bir yöntemdir. Bu yöntem, kullanılan veya işaretlenen bloklar için adına Dosya Ayırma Tablosu (FAT32) denen bağlantılı liste veri yapısını kullanır. Bu iki yöntem farklı avantaj ve dezavantajlara sahiptir. Biz sistemde daha iyi bir hybrid çözüm üretmek için her iki yöntemi de kullanıyoruz. Kümelerin boş olup olmadığını görmek için başlık kısmındaki bitmap yapısı kullanılmaktadır. Dosya izini takip etmek için liste tabanlı FAT yapısı kullanılmaktadır ve bu kavramlar ileri ki bölümlerde açıklanmıştır.

1.4.6. Programlama Dilleri

Bu projede Microsoft Visual Studio 2017 frameworku kullanıldı. Ayrıca, tüm sistemi uygulamak için C # programlama dili tercih edildi.

2. MATERYAL VE YÖNTEM

2.1. EBYS Tasarımı

Çalışmamızda yukarıda belirtilen EBYS'nin önemi, faydası ve kullanılma gerekçelerine uygun olarak EBYS geliştirilmesi çalışması yapılmaya başlanmıştır. Bu sayede big data test ortamı oluşturulmaya çalışılmıştır.

Yazılım modellemesi için aşağıdaki geliştirme aşamaları uygulanmalıdır.

- Planlama
- Analiz
- Form Tasarım
- Programlama
- Test ve hataların düzeltilmesi

İlk önce evrak işlemlerinin bir kurumda nasıl hareket ettiği ile ilgili analiz çalışması yapılmıştır. Yetkili kişilerle görüşülerek evrak sisteminin çalışma şekli incelenmiştir.

Modelleme için yazılım hazırlanırken kullanıcı ara yüzünün kolay kullanılabilir, yapılacak işlemlerin kolay yapılabiliyor olmasına dikkat edilmiştir. Her seviyede ki personel için anlaşılır olması sağlanmaya çalışılmıştır.

2.1.1. EBYS'nin Çok Katmanlı Mimari Yapısı

• User Interface: Kullanıcı Arayüzleri

Kullanıcıya sunulan ara yüzlerdir., Login Ekranı, Evrak Kayıt Ara yüzü, Gelen Evrak Ara yüzü, Evrak İşlemleri Ara yüzü (paraf, havale), Yeni Evrak Ara yüzü, Yetki Ara yüzü, Vekalet Ara yüzü gibi.

• Business Layer: İş Katmanı

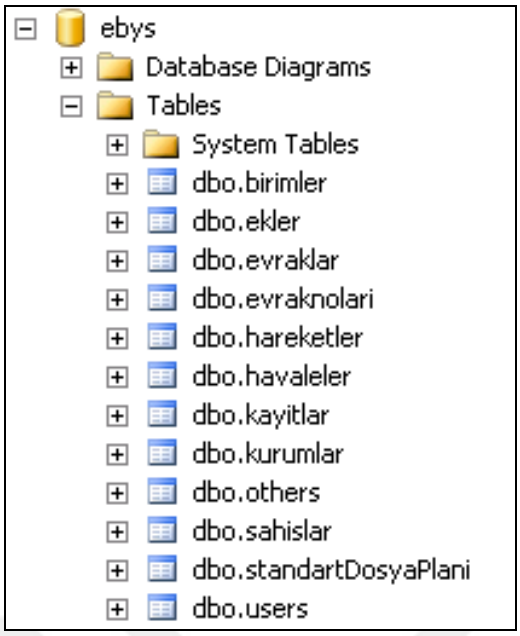
Ara yüzlerdeki tetiklemeler veya işlemler sonucu devreye giren class, method, property'lerdir.

• Data Access Layer: Veri Erişim Katmanı

Select, Insert, Update, Delete gibi SQL cümleciklerinin koşturulup sonucunun döndürüldüğü katmandır.

2.1.2. Veri Tabanı Tasarımı

Bu sistemle ilgili veri tabanı tasarımı yapılmıştır. Veri tabanı dosyası, kurumdaki birimleri gösteren birimler tablosu, evraklara ek olarak ilave edilen kayıtların tutulduğu ekler tablosu, evrak numaralarının tutulduğu evraknolari tablosu, oluşturulan evrakların kimler üzerine zimmetlendiğinin kayıtını tutan hareketler tablosu, evrak kayıt işleminin tutulduğu kayıtlar tablosu, gelen/giden dış evrakların kurumlarının kaydedildiği kurumlar tablosu, kurumda çalışmayan diğer kişilerin tutulduğu sahislar tablosu, standart dosya planının tutulduğu standartDosyaPlani tablosu ve kullanıcı kayıtlarının tutulduğu users tablosundan oluşmaktadır. EBYS yazılımında kullanılacak olan veri tabanının tablo (table)'ları Şekil 2.1'de gösterilmiştir.



Şekil 2.1. EBYS Veri Tabanı Tabloları

Veri tabanı tasarlanırken gereksiz veri tekrarı ve veri kaybı olması önlenecek şekilde tasarım yapılmıştır. Normalizasyon işlemi yapılarak tabloların, alanların ve ilişkilerin normal forma gelmesi sağlanmıştır. Bu sayede, veri tabanında daha sonra, veri ekleme, silme, güncelleme gibi işlemlerden sonra oluşabilecek veri sapmalarını engellemek için normalizasyon işlemi yapılmıştır. Özellikle bir veri tabanı içinde yer alan ilişkisel durumlarda bir tabloda bir veri değişimi sonucunda kendisi ile ilişkili olan diğer tablolarda da gerekli değişikliklerin olması gibi birçok durum, düzenli bir veri tabanı tasarımını gerekli kılmaktadır.

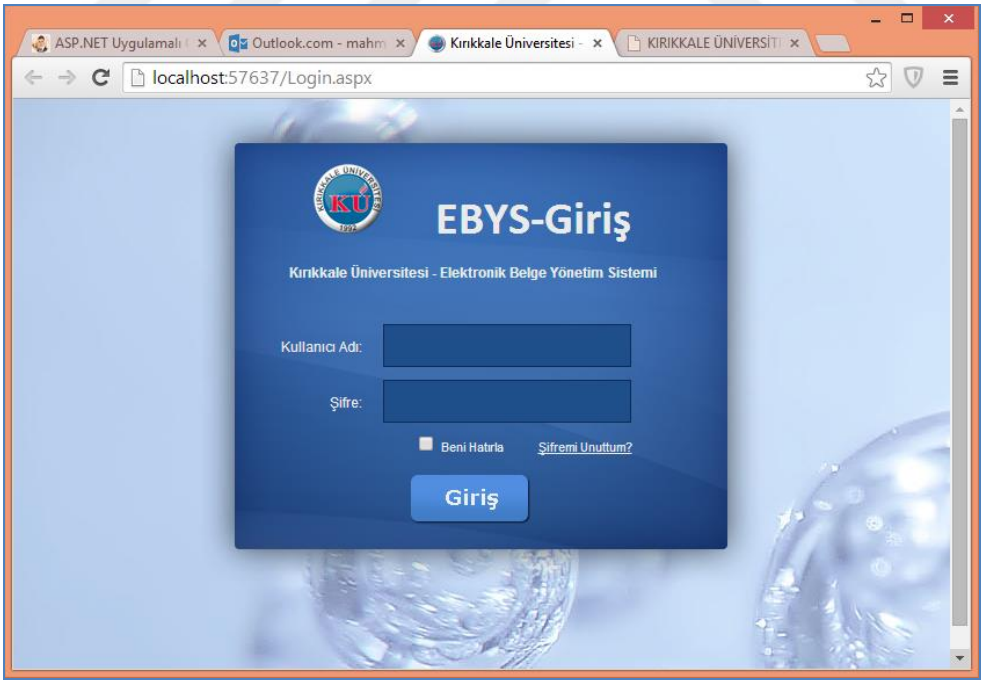
Oluşturduğumuz veri tabanı içindeki tüm tablolar verilerimizi saklayacak ve bu verilere gerektiği anda ulaşabilecek, bunları değiştirebilecek ve ilişkisel olarak tablolarımızı okuyabilecek şekilde tasarlanmıştır. Oluşturulan veri tabanında normalizasyon işlemi gerçekleştirilmiş, ilişkiler rafine edilmiş, veri kaybı, veri tekrarı ve veri yetersizliği gibi olumsuzluklar ortadan kaldırılmış ve en az satır ve sütun kullanılarak tekrardan arındıran alt kümelere ayırma işlemi yapılmıştır.

Veri tabanı tasarımı yapıldıktan sonra EBYS modelleme tasarımı yapılmaya başlanmıştır.

2.1.3. Modellemenin Form Tasarımları

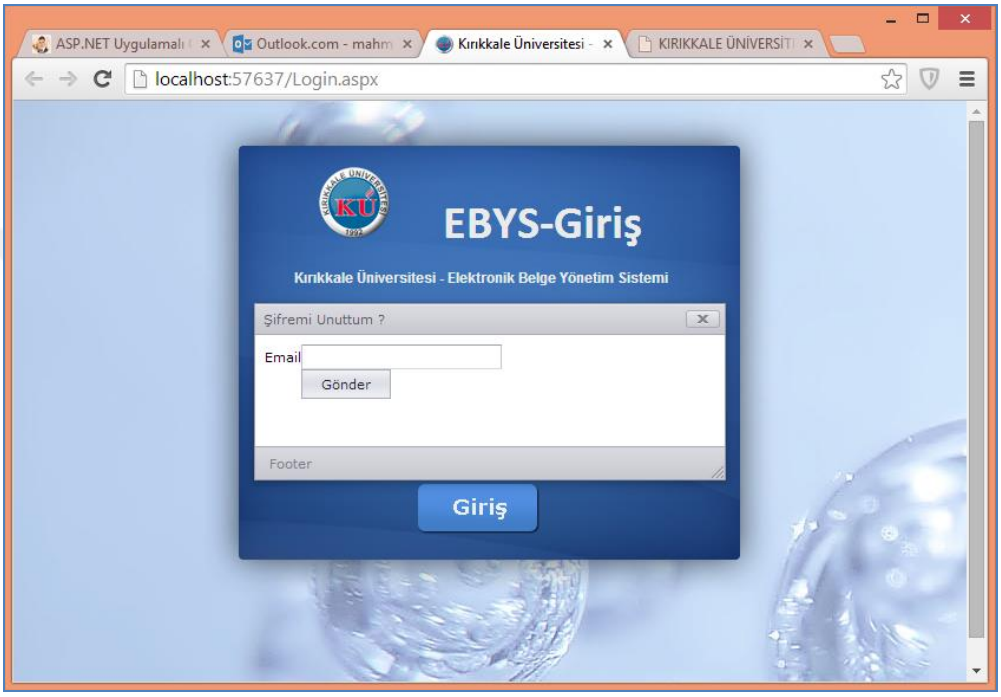
Oluşturulan EBYS'nin bazı formları ve kodları aşağıda gösterilmiş olup formaların kod görüntüsü tezin ekler bölümünde bulunmaktadır.

Sistemde yeni kullanıcı güvenliği sağlamak için kurum içi çalışan personellerdir. Kullanıcı başvurusunu ilgili birime yapan personele kullanıcı adı ve şifresi gönderilir. Giriş ara yüzünde, kullanıcı adı ve şifre dışında, tekrar girişte kullanıcı adının hatırlanabilmesi için beni hatırla onay kutusu ve unutulan şifrenin hatırlatılması için şifremi unuttum linki bulunur. Giriş ara yüzü form görüntüsü Şekil 2.2'de gösterilmiştir.



Şekil 2.2. Kullanıcı girişi ekranı: login.aspx.

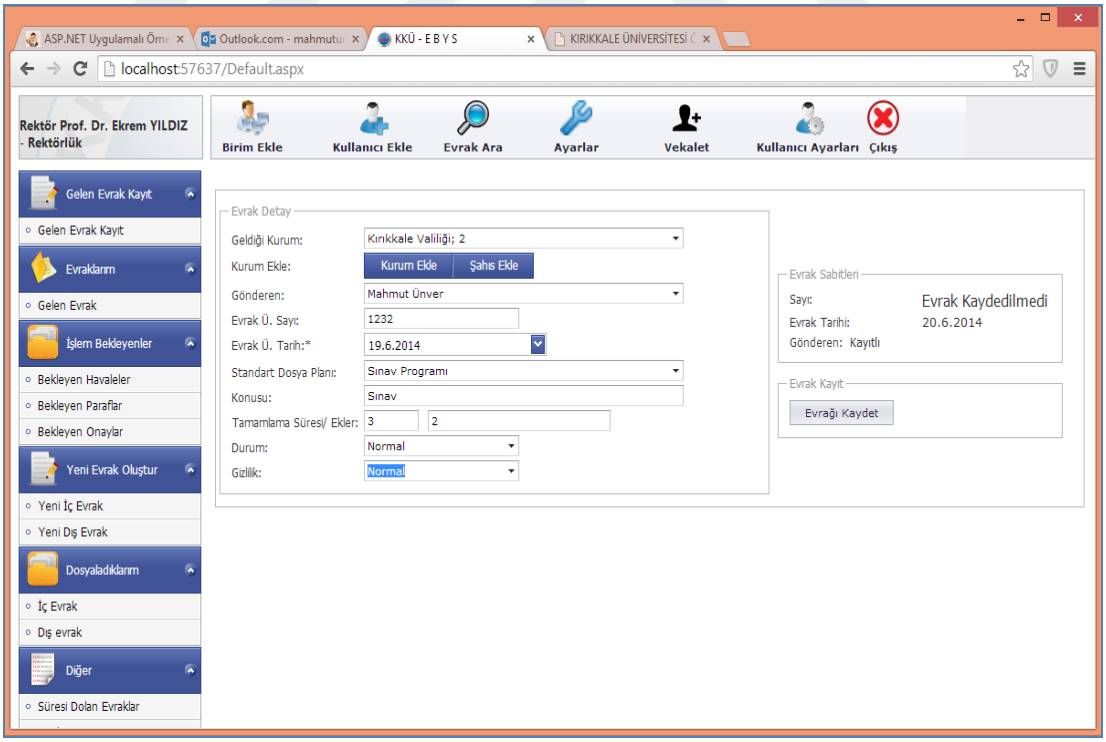
Şifresini unutan kullanıcılar için şifremi unuttum ara yüzü Şekil 2.3'de gösterilmiştir. Kullanıcı adı ve elektronik posta adresi veri tabanı tablosunda karşılık gelen kullanıcının e-posta adresine yeni şifre gönderilmektedir.



Şekil 2.3. Şifremi unuttum ekranı: SifremiUnuttum.aspx.

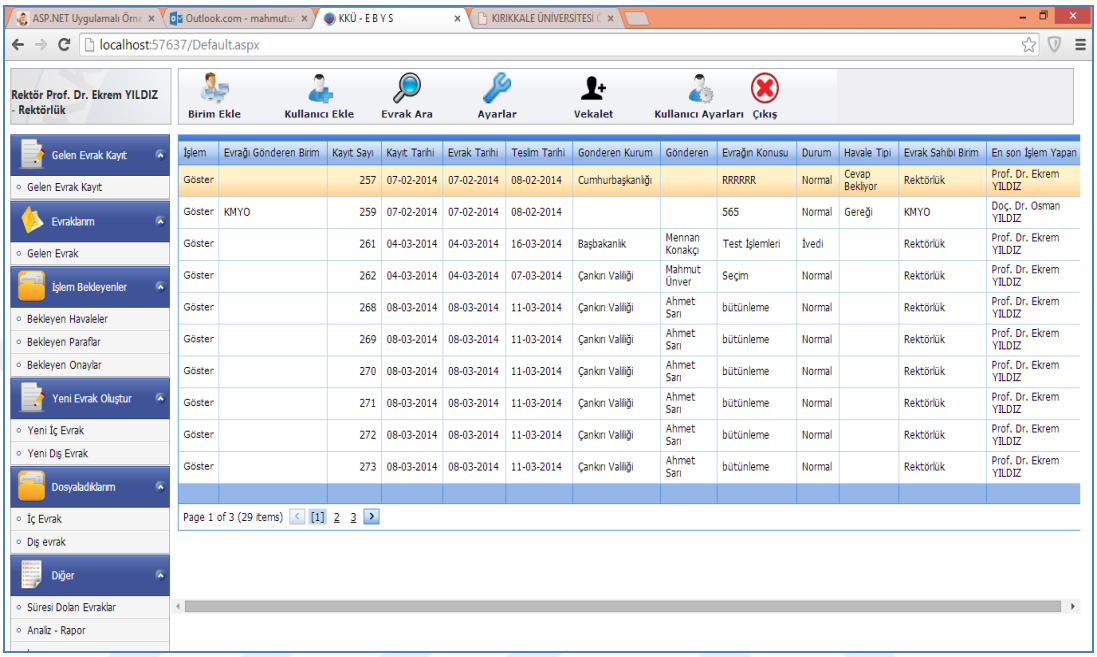
Kullanıcı doğru kullanıcı adı ve şifre ile oturum açtığında Şekil 2.4'te gösterilen ana sayfa açılmaktadır. Kullanıcı dostu bir tasarım ile kullanıcı yapmak istediği işleme kolay anlaşılabilir menüler sayesinde ulaşabilmektedir.

Şekil 2.4. Ana sayfa ekranı: default.aspx.



Şekil 2.5. Gelen Evrak Kayıt Ekranı: yeniEvrakKayit.aspx.

ekranaYaz(), gidenEvrakKaydet(), gecenSure() metotlarının yazıldığı yeni evrak kaydet ara yüzü (Şekil 2.5) ile formda görülen bilgiler ile yeni evrak kaydedilecek ve oluşturulan kişiye zimmetlenecektir. Evrak Kaydet butonu ile evrak kaydedilecektir.



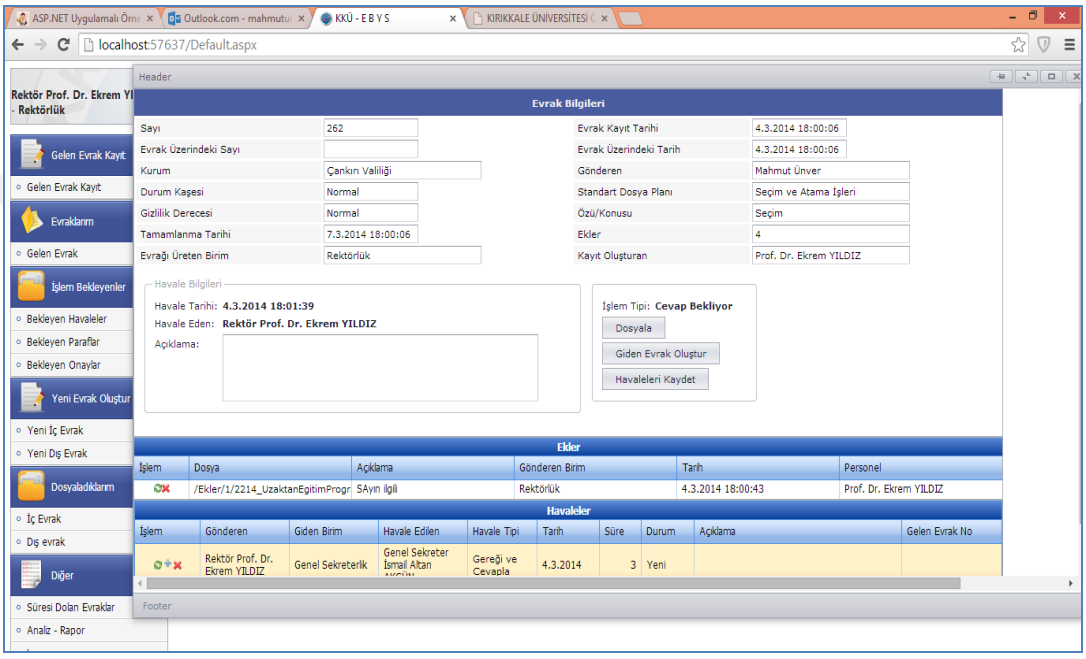
Şekil 2.6. Gelen evraklar listeleme ekranı: gelenveraklar.aspx.

Şekil 2.6'da gösterilen Gelen Evrak Listeleme formu ile kullanıcının üzerinde olan iç ve dış bütün evraklar listelenmektedir. Formdaki ilgili evrakın satırında göster butonu tıklandığında o evrakla ilgili Şekil 2.7'de gösterilen yapılacak işlemler sayfası açılmaktadır.

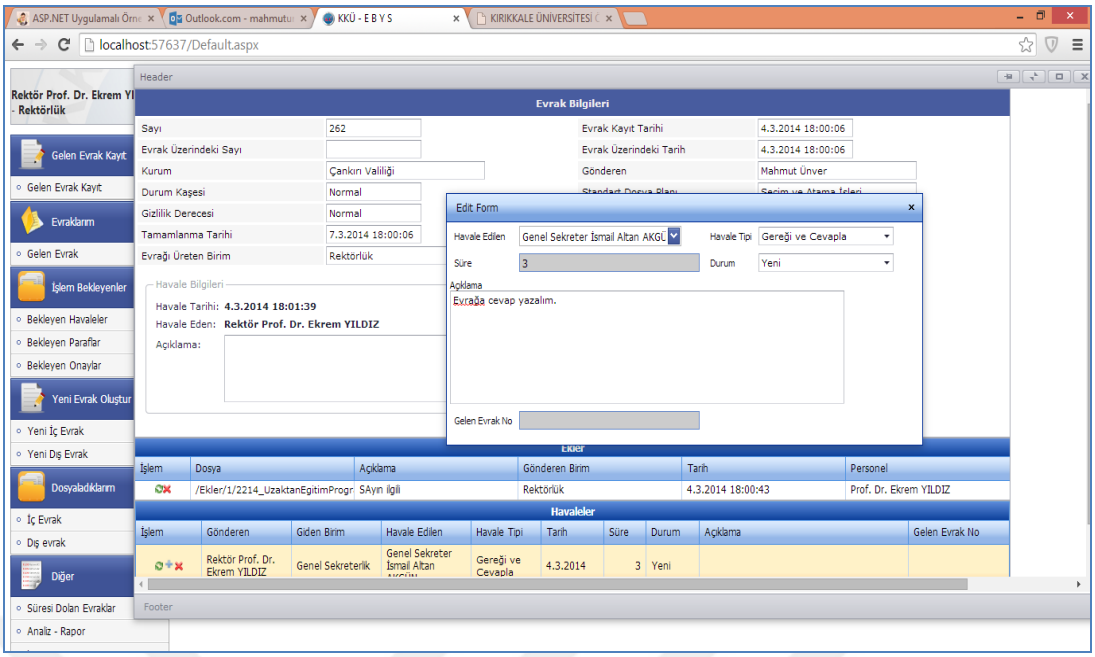
Havale yetkisi olan bir personel için evrak havale penceresi aynı form üzerinde açılacaktır. Kayıtlı birimlerdeki kayıtlı personele kullanıcının hiyerarşik olarak havale edebileceği kişileri kontrol ederek havale işlemi Şekil 2.8'de gösterilen arayız yoluyla yapılabilmektedir. Şekil 2.9'da gösterilen havaleler ara yüzü ise kullanıcının üzerinde işlem yapılmayı bekleyen havalelerin listesini göstermektedir. Aynı şekilde

Şekil 2.10'da aktif kullanıcıya yapılmış ve kullanıcı tarafından işlem görmemiş gelen paraflanacak evraklar listesi listelenmektedir.

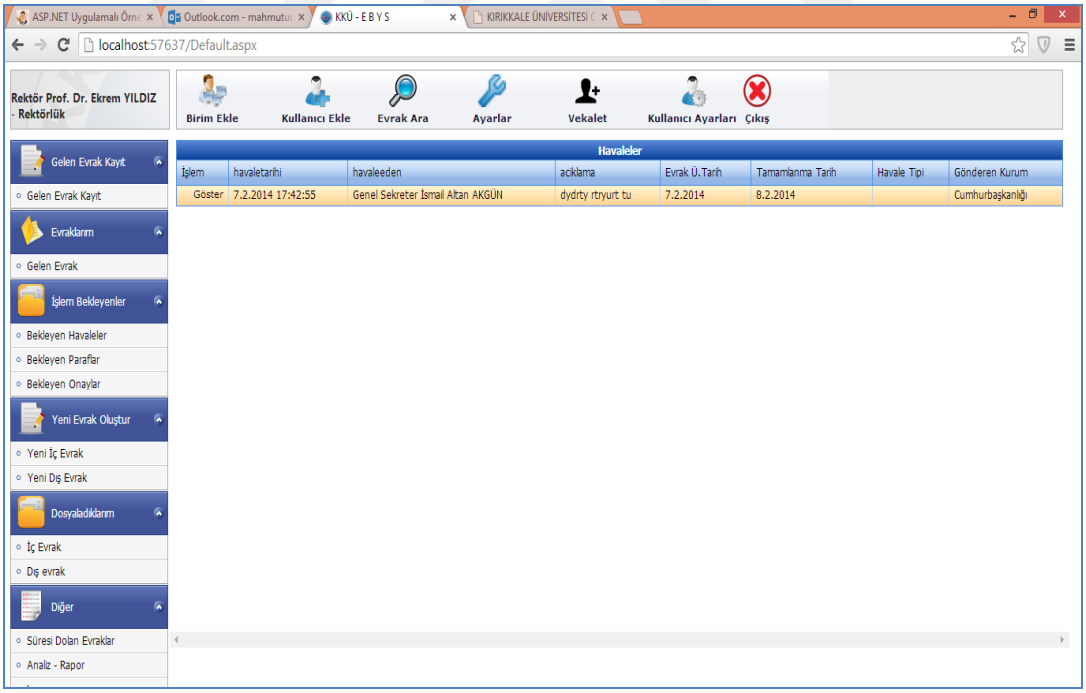
Yetkisi olan kullanıcılar için Şekil 2.11'de kurum içi ve kurum dışı yeni kurum ekleme işlemi yapılmaktadır. Şekil 2.12'de yeni evrak oluşturma ara yüzü tasarlanmıştır.



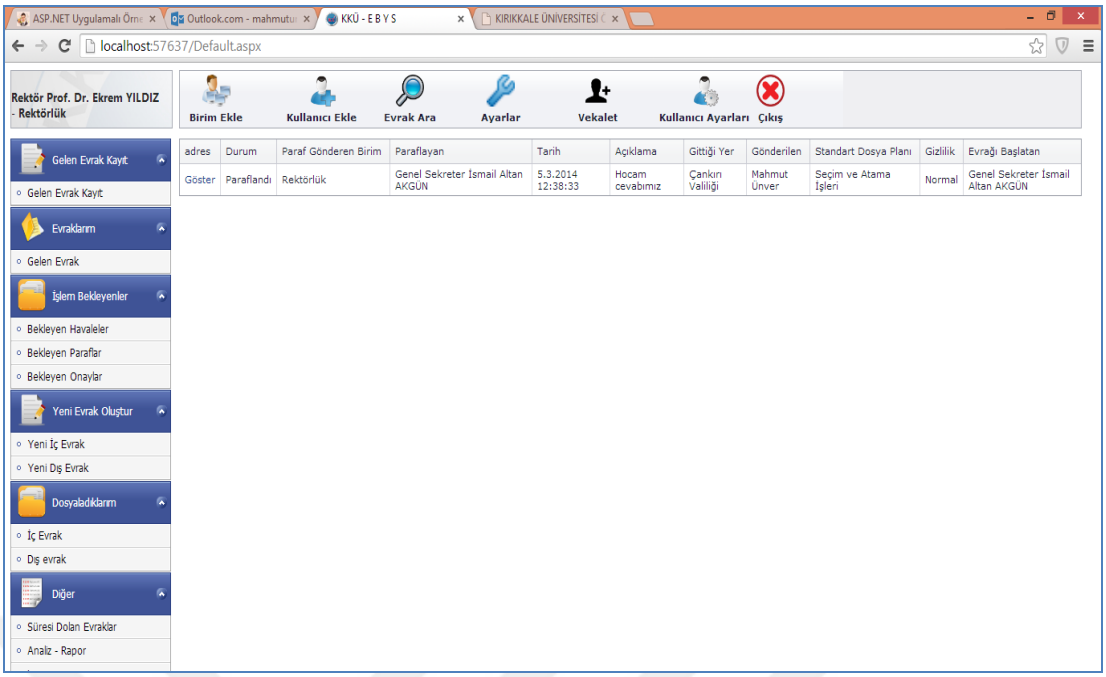
Şekil 2.7. Gelen evrak bilgileri gösterim ekranı: gelenevrakyonet.aspx.



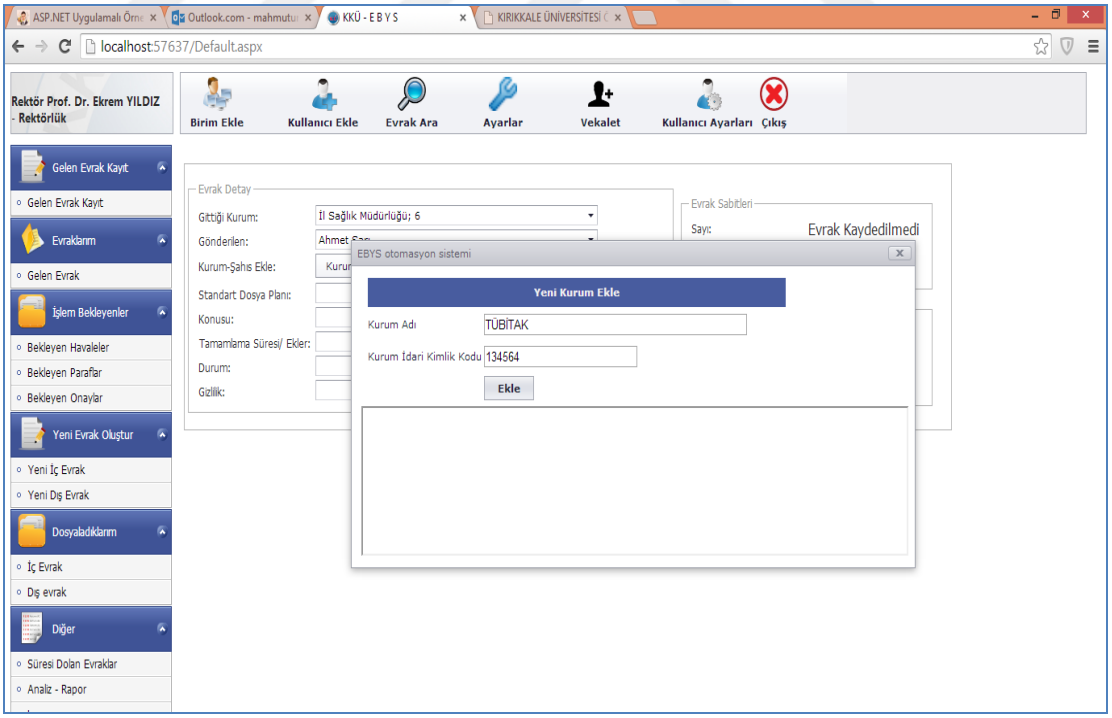
Şekil 2.8. Evrakı havale etmek.



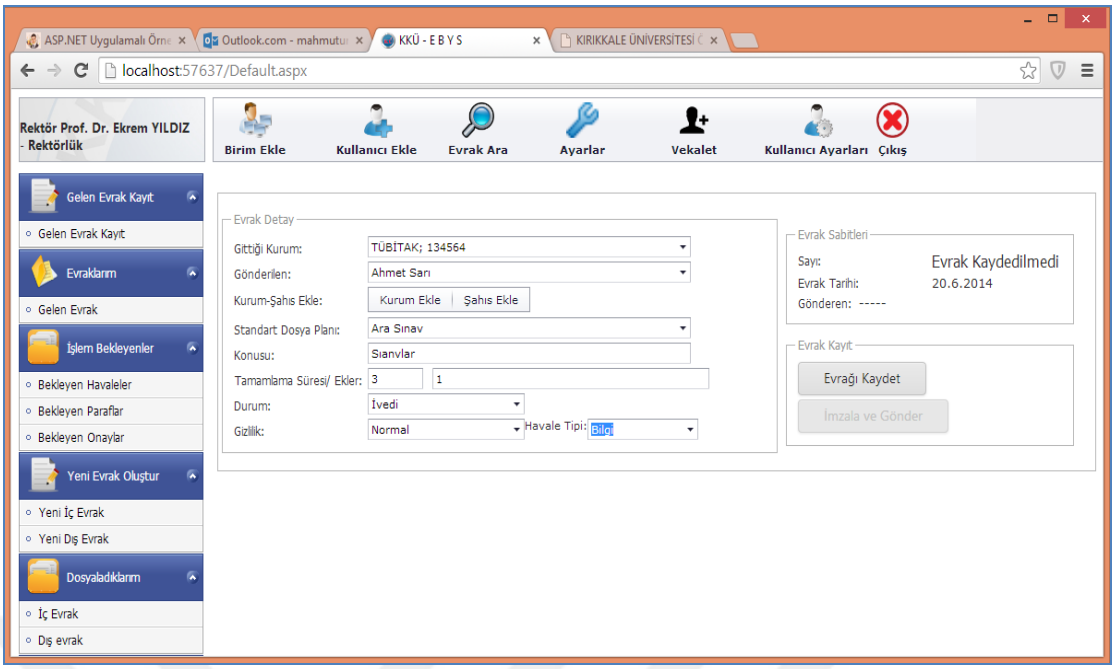
Şekil 2.9. Bekleyen havaleler ekranı: havaleler.aspx.



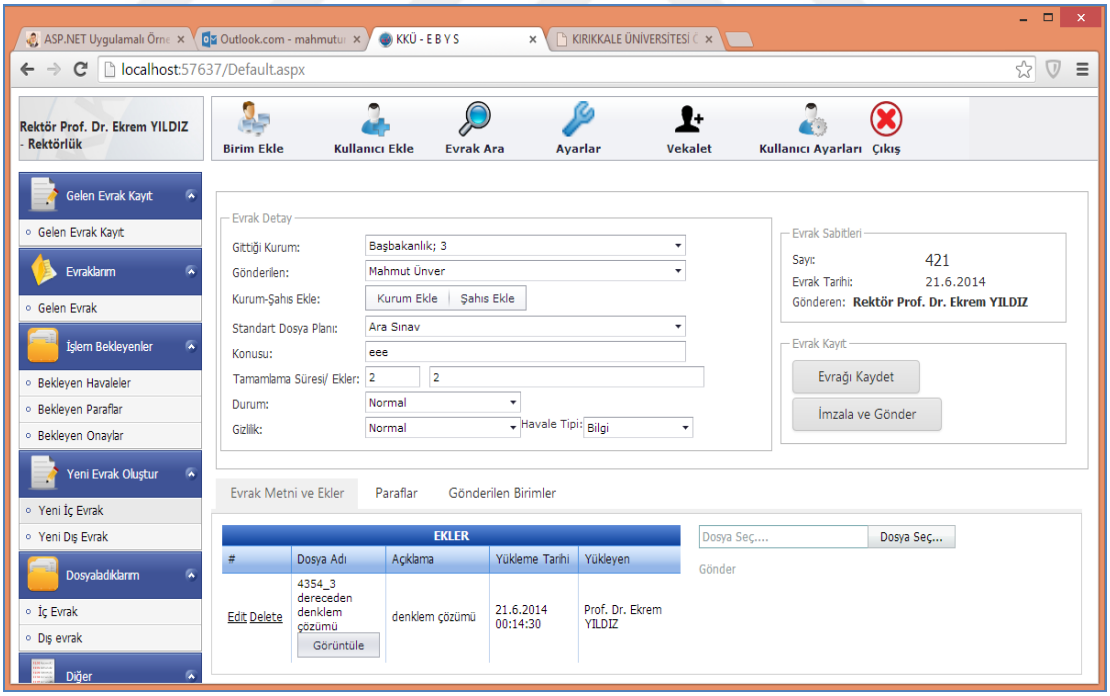
Şekil 2.10. Bekleyen paraflar ekranı: paraflariver.aspx.



Şekil 2.11. Yeni kurum ekle ekranı.



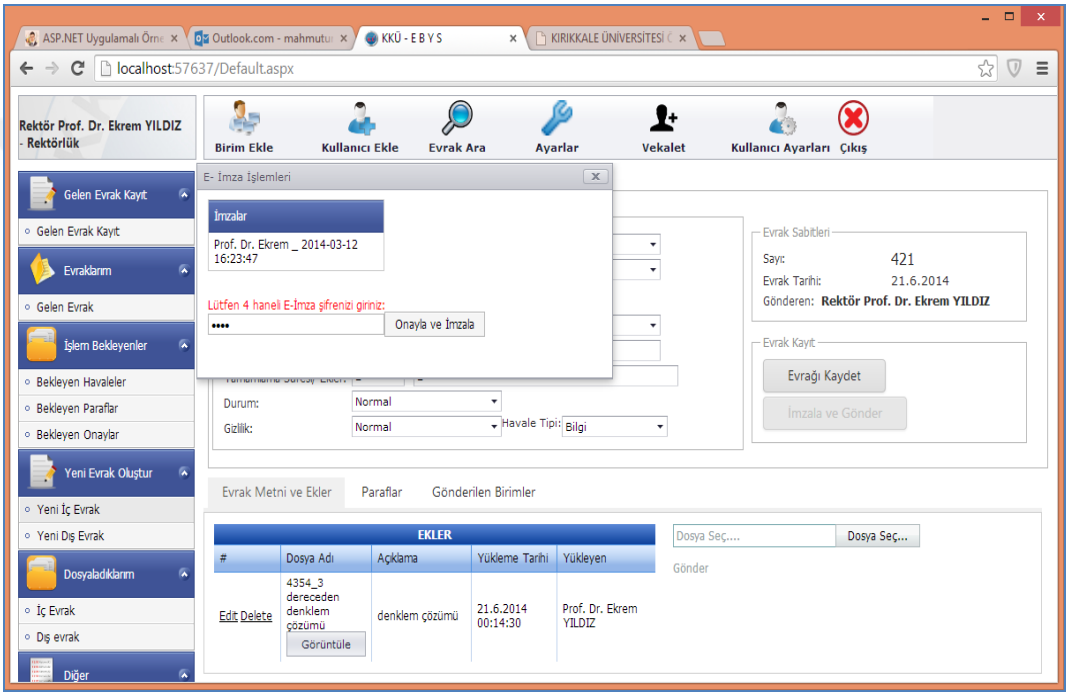
Şekil 2.12. Yeni evrak oluşturma ekranı: Default4.aspx.



Şekil 2.13. Kaydedilmiş evrak ekranı.

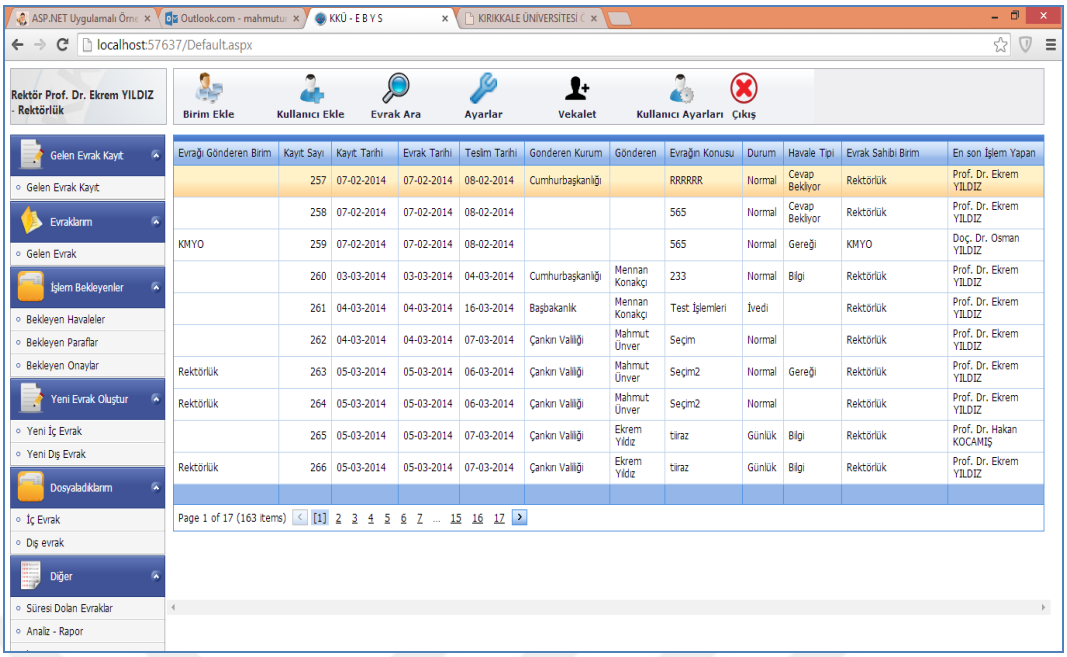
Yeni evrak oluşturulduktan sonra, kaydedilmiş olan evraklar ve o anda zimmetlenmiş olduğu birim yetkilisi Şekil 2.13'deki ara yüzde gösterilmektedir. Bu aşamada yeni oluşturulan evraka evrak kayıt numarası ve tarihte verilmektedir.

Şekil 2.14'deki form görüntüsü evrakın elektronik imza ile imzalanması aşamasını göstermektedir. Kullanıcı e-imza donanımını ve e-imza şifresini kullanarak evrakı imzalamaktadır.

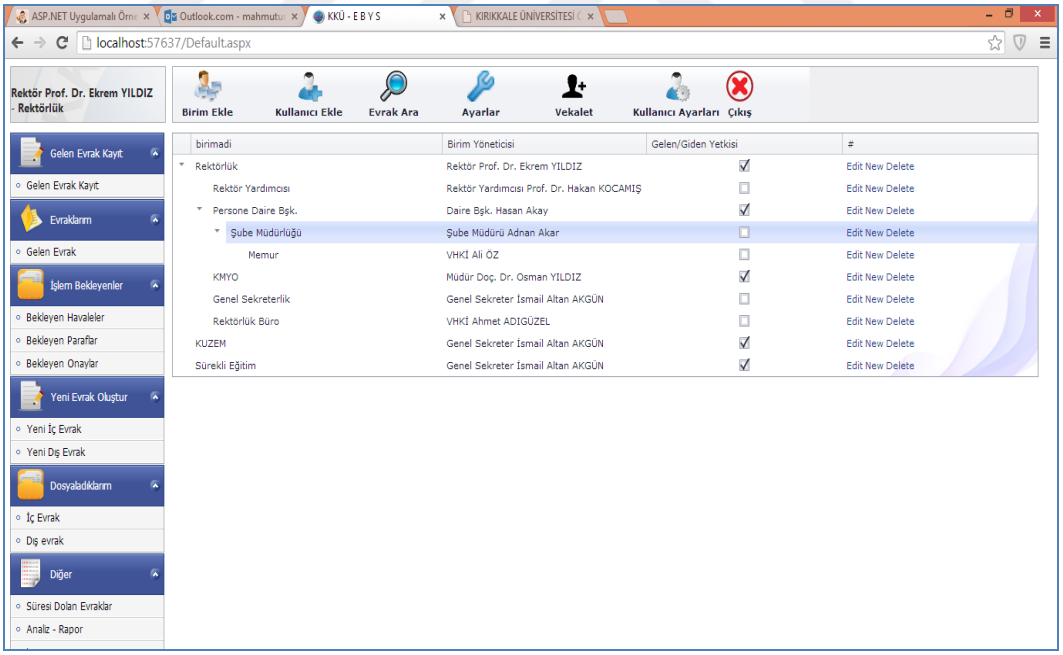


Şekil 2.14. e-imza ile imzalama ekranı.

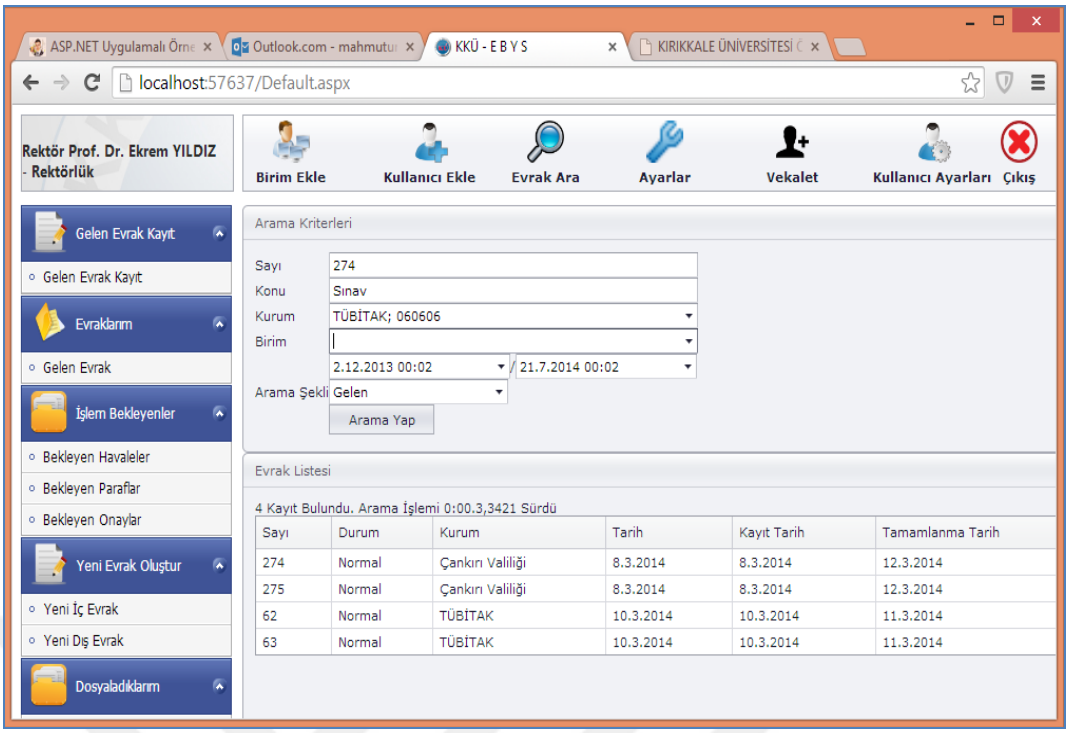
Evrakların günlü olup olmamasına bağlı olarak kullanıcılar için üzerindeki evraklara işlem yapma süreleri vardır. Şekil 2.15, süresi dolan evrakları göstermektedir. Kurum içerisinde yeni birim ekleme ara yüzü (Şekil 2.16) treelist formatında hazırlanmıştır.



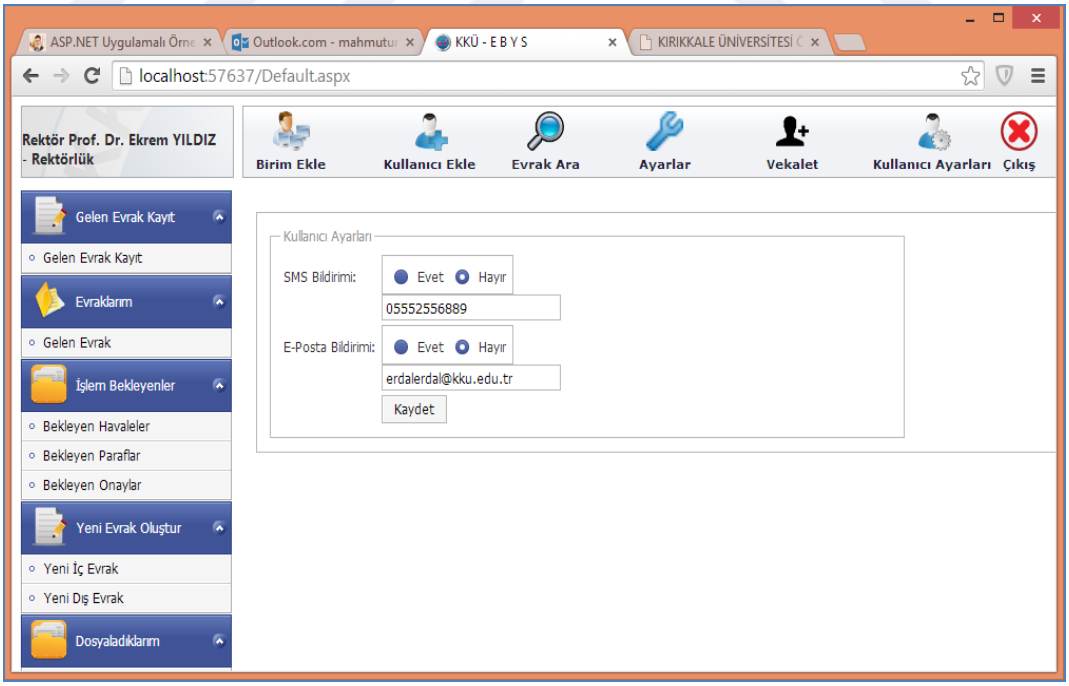
Şekil 2.15. Süresi dolan evraklar listesi: SuresiDolanEvraklar.aspx.



Şekil 2.16. Hiyerarşik (treelist) birim ekleme ve yönetim ekranı: birimDegistir.aspx.



Şekil 2.17. Evrak arama ekranı: Arama/EvrakArama/Default.aspx.



Şekil 2.18. Kullanıcı bildirim ayarları ekranı: userSettings.aspx.

EBYS yazılımlarının günümüzde kullanılmalarının önemli etkenlerinden biri evrak arama işleminin daha kolay yapılabiliyor olmasıdır. Tasarlanan EBYS'de evrak aram kolay bir şekilde yapılmakta ve en doğru sonucu verecek şekilde planlanmıştır. Evrak arama ara yüzü Şekil 2. 17'de gösterilmiştir. Kullanıcıların kişisel profillerinde değişiklik yapılabilmektedir. Sms ve e-posta ayarları Şekil 2.18'de gösterilmiştir.

2.2. Dosya Sistemi Tasarımı

Bu çalışmada big data setleri için hızlı, güvenli, sunucusuz, sağlam, dayanıklı, kolay yönetilebilir ölçeklendirilebilir dağıtık dosya sistemi geliştirilmiştir. Bu sistem öncelikli olarak Kırıkkale Üniversitesi için geliştirilmiş bir Akıllı Dağıtık Dosya Sistemidir (KÜADDS). KÜADDS yapısal olarak Hadoop, NFS ve AFS gibi büyük miktarlardaki veriyi yönetmek için tasarlamıştır. Bu çalışmanın farklılıkları aşağıda detaylı bir şekilde anlatılmıştır.

Güvenlik önlemleri sayesinde, geliştirilen orta katman platformunun güvenliği önemli ölçüde artmıştır.

Elektronik belge yazılım sistemleri, işlem yapmaları için sunucudan işlem için ne kadar süre bekleyeceklerini tolore edememektedirler, bu nedenle tüm işlemler mümkün olan en kısa sürede veya kabul edilebilir bir zaman aşımı süresi içinde çalıştırılmalıdır. Hadoop ve diğer çözümler bu sorunu rahatlıkla çözebilmeleri için çok fazla yükleri var. Çünkü bunlar ağırlıklı olarak big data depolama ihtiyaçları için oluşturulmakta ve sadece performansta iddialı olmak değil, MapReduce frameworku yoluyla grid hesaplama işlemlerinin üstesinden de gelmek tasarlanmışlardır. Sonuç olarak, Kırıkkale Üniversitesi elektronik belge yazılımı için, kendi dağıtık dosya sistemini geliştirme ihtiyacını gidermiş olacaktır. Bu da çalışmamızın en önemli unsuru olacaktır.

Sistemin performansını değerlendirmek için diğer veri işleme yöntemleri ile karşılaştırmalar yapılmıştır. Karşılaştırma sistemlerinin özellikleri:

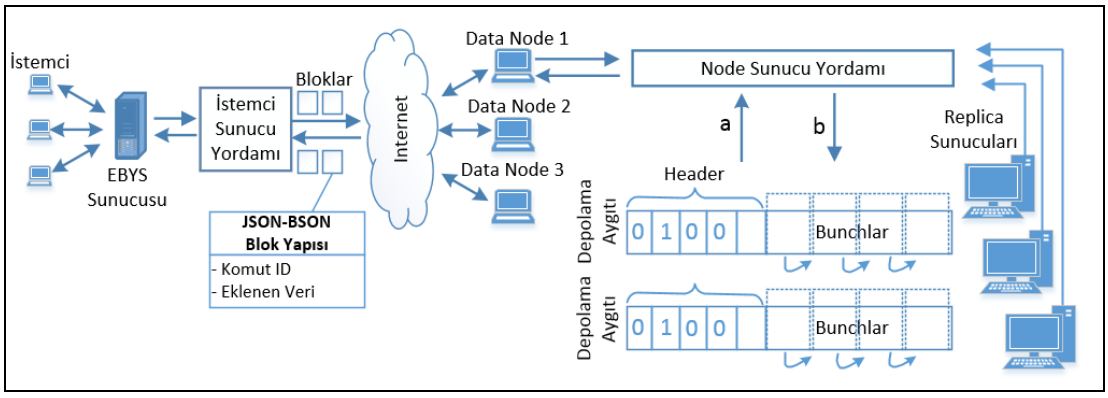
Hadoop:
Karşılaştırmak için kullanılan Hadoop konfigürasyonu bir adet name node ve üç adet data node içerir. Red Hat Enterprise Linux Server 6.0, her düğümde çalışır. Ayrıca, her bir düğümde Java-1.6.0, MongoDB-3.4.2 ve Hadoop-2.7.2 yüklüdür.

CouchBase:
Karşılaştırmak için kullanılan sistemlerden biri de popüler bir NoSQL veri tabanı sistemi olan Couchbase'dir. İncelenen makalede [71], Elizabeth Gallagher'ın yaptığı çalışma referans gösterilerek, Couchbase'in diğer popüler NoSQL veri tabanları kadar güçlü olduğu iddia edilmiştir. Sistemde her bir bucket 20 MB kümeden oluşur. Test makinesinde 6 GB RAM ve 200 GB depolama alanı vardır. Microsoft SQL Server 2014 ayrıca İlişkisel Veri Tabanı Yönetim Sistemleri veri tabanı motoru ile karşılaştırmalar yapmak için aynı makinede seçilip kurulmuştur.

Sistem bileşenleri, istemci uygulamaları, data node'lar, sanal dosya sistemi ve replica node'lardır.

2.2.1. Sisteme Genel Bakış ve Entegrasyon

Elektronik belge verileri son yıllarda son derece artmıştır. Günümüze kadar, log kayıtları, belgenin kendisi, dağıtım ve havale edilmesi yoluyla kopyaları, belgelerin üst verileri, pdf formatları, belgelerin eklerinin resim formatları gibi farklı elektronik belge verisi oluşturulmuştur. EBYS bu dosyaları raporlama, havale, gösterme, veri aktarımı, belge takibi, gibi amaçlarla kullanmaktadır. Bu sayede evrak bulma ve izleme işleme daha iyi yapılabilmektedir. Bu çalışmada, istemci uygulaması, statik IP'ye sahip bir web sunucusunda çalışan bir elektronik belge yönetim sistemini ifade etmektedir.



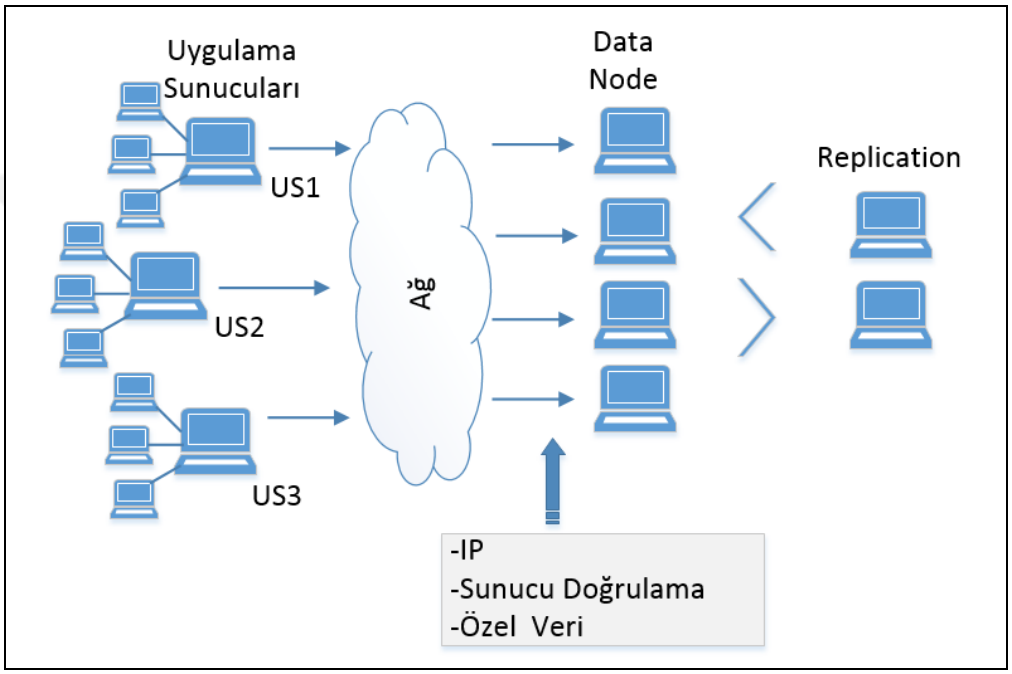
Şekil 2.19. Windows hizmet yordamları ve JSON paketleri.

Bu çalışmada kullanılan önemli teorik ve kavramsal frameworkler vardır ve bunlar üç başlık altında özetlenebilir: servis-yordamları (İstemci ve sunucu tarafı), güvenlik sorunları (güvenli bağlantı için) ve dağıtık dosya sistemi mimarisi (sistemimizin ana bölüm). Servis yordamları, istemci tarafı ve sunucu tarafı arasında ve istemci sunucu yordamı (Client-Server Routine-CSR) ve data node servis yordamı (data-node service routine-DNSR) arasında güvenli iletişimi sağlamak ve için istemci-sunucu soket araçları kullanılarak geliştirilmiştir. Bu servisler, Şekil 2.19'da JSON veri yapısında TCP / IP üzerinden gösterilen iletim blokları kullanılarak birbirleriyle iletişim kurar. İstemci uygulamasının sisteme entegre edilebilmesi için istemci uygulamasında DLL dosyaları yüklü olmalıdır. CSR, EBYS dosyalarının boyutuna ve türüne bakılmaksızın DNSR'ye göndermek ve talep üzerine okumaktan sorumludur. CSR, istemci uygulamasının isteklerini (EBYS) güvenli bir bağlantıyla DNSR'ye gönderir. Ayrıca, dosyaları aramak için replika node'ları için bir adet daha küçük Windows servisi uygulanmıştır. Dahası, bu servis diğer node'larla şifrelenmiş veri aktarımı sağlar.

Sunucu tarafı bu çalışmanın önemli bir parçasıdır, aşağıdakilerden sorumlu kapsamlı bir çekirdek servisidir. a) Herhangi bir istemci uygulama talebi için belirtilen port numarasının dinlenmesi b) Kimlik doğrulama süreci CSR IP değerini kontrol etmeyi içerir. c) Okuma, yazma ve isteğe göre işlemleri silme.

Çalışmanın bir diğer önemli noktası ise aşağıdakileri içeren güçlü bir güvenlik seviyesine ulaşılmış olmasıdır: a) Birincil olarak CSR'nin statik IP değeri b) Veri aktarımı için JSON ve BSON veri yapısı c) Simetrik şifreleme algoritması.

2.2.2. İstemci Uygulamaları



Şekil 2.20. Sistem Genel Yapısı

İstemci uygulamaları KÜADDS sisteminden dosyalama hizmeti alan uzaktaki uygulamalardır. Uygulamalar tüm isteklerini TCP/IP üzerinde yapacağından platformdan bağımsızdır. Bunların statik bir IP'ye sahip olması yeterlidir. İstemci uygulamaları dosyalama ve arşivleme amaçlı olarak "write once-read many" şeklinde bu sistemi kolaylıkla kullanabilmektedir. KÜADDS'den dosyalama hizmet almak isteyen uygulama, bağlantı için Dynamic Link Library'yi kendi sistemine kurmalıdır. Bu sistemde, GetAvailableDatanodeIp, ChangePassword, SaveFile, ReadFile, DeleteFile, GetFileFromReplica metotları mevcuttur ve sonraki bölümlerde

detaylandırılmıştır. İstemci uygulamaları kendisine rezerve edilen KÜADDS'deki data node'lar ile iletişime geçer ve tüm işlemlerini bu node ile yönetir. İstemci uygulaması kendilerine simetric kriptoloji ile gönderilen kullanıcı adı, şifre ve sahip olduğu statik IP ile kendilerine ayrılan data node'a bağlanır. İstemci uygulamalarının sayısında herhangi bir sınır olmadığı gibi bu uygulamalar KÜADDS üzerinde tanımlı sunucu node'lardan birden fazlasını kullanabilmektedir. Bu, sistemimizin diğer sistemlerde olmayan bir özelliğidir. Sistemimizin genel yapısı Şekil 2.20'de gösterilmektedir.

2.2.3. Nodelar

Sistemin en önemli bileşenlerinden birisi node elemanlarıdır ve iki tip node vardır: Bunlar, Data node ve replica node'tur. Dağıtık sistemlerde genellikle Head-Name node ve data node kullanılmaktadır. Bu çalışmada head node'un tüm işlevlerini data node üzerine eklemek yoluyla sadece data node kullanılmıştır. Bu node'lar hem data hem de server node gibi çalışabilen oldukça fonksiyonel bir yapıdadır. Bu sayede sistemin sürdürülebilirlik ve dayanıklılık özellikleri çok iyi duruma getirilmiştir. Tüm sistemde sadece bir node kalsa bile sistem çalışmasına devam edebilmektedir. Diğer sistemlerde head node çalışmadığı zaman bütün sistem erişilemez olmaktadır. Bu da sistemimizin sahip olduğu özelliklerden biridir.

2.2.4. Data Node

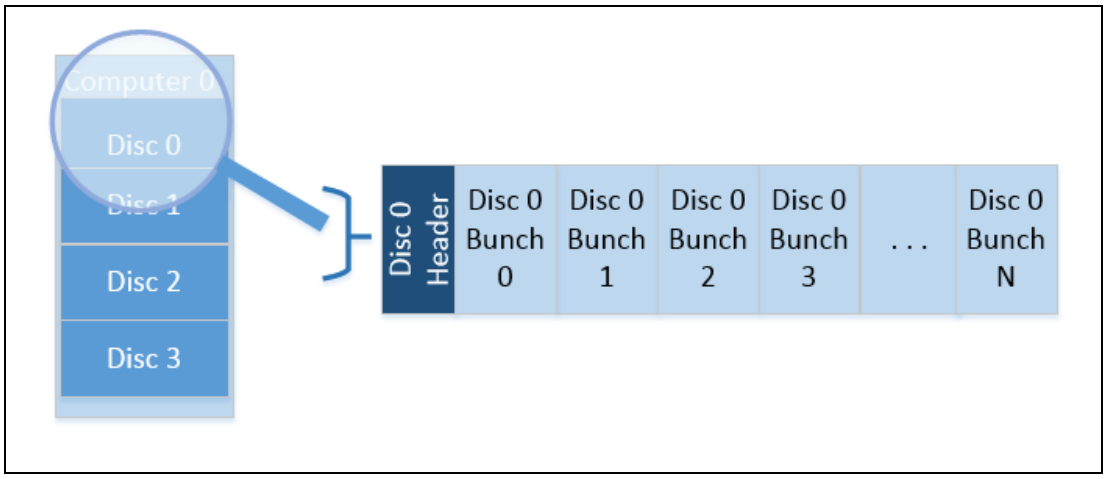
Data node'lar, bilindiği gibi, istemciden gelen dosyaları saklamak ve yönetmek için kullanılmaktadır. Hizmet alan istemci uygulamaları, data node'lar ile TCP/IP ağ servisi aracılığı ile iletişime geçer. Bu çalışmada özellikle Data node'ların her biri farklı uygulamalara hizmet veren aynı seviyedeki sunuculardır. Yazılım, bizim tarafımızdan geliştirilen dynamic link library, data node'lara Symmetric-key algoritması ile güvenli bir bağlantı sağlamaktadır. Bununla birlikte uygulama sunucuları, data node'lardan hizmet alan, sabit IP adresine sahiptir. Bu IP üzerinden

data node'lara bağlanır. Data node'lar ayrıca kendilerinden hizmet talep eden isteklerin IP adresini de kontrol etmektedir. Bunlara ilaveten istemci, istediği durumda, data node'lara gönderdiği veriyi symmetric key algoritması ile şifreleyerek güvenli şifrelenmiş dosya olarak gönderebilir. Bu sayede hem statik IP hem de şifrelenmiş dosya sayesinde güçlü güvenlik seviyesine ulaşılmıştır. Data Node'lar istemciden gelen istekleri kendi üzerlerinde tanımlı olan aslında veri depolama aygıtı üzerinde büyük bir dosya olan Sanal Dosya Sistemi üzerinde saklayacaktır. Sistemin en önemli özelliklerinden biri geliştirilen sanal dosya yapısıdır. Ayrıca, her bir data node birden fazla istemci uygulamasına hizmet verebilmektedir. Node'lar, en uygun yükleme dengesini sağlamak için sırayla dosya depolamak için kullanılırlar. Ayrıca istemci uygulamalarının her biri birden fazla data node'a bağlanabildiği gibi her bir node da birden fazla istemci uygulamasına hizmet verebilmektedir. Bu özellik sayesinde sistemin hem ölçeklenebilirliği hem de kullanılabilirliği iyi olmaktadır.

2.2.5. Sistem Servis Mimarisi

Sistemin sahip olduğu tüm işlevlerin yerine getirilmesi için her bir data node'ta çalışan bir servistir. Bu program, sunucu soket yapısını kendisine tahsis edilen port (443 HTTPS için ayrılmıştır, ancak biz de bu portu kullandık) aracılığıyla dinler. Servis yordamı etkinleştirildiğinde:

1. Bilgisayardaki tüm veri depolama cihazlarını tanımlar ve cihaz listesini hazırlar.



Şekil 2.21. Sistem veri depolama mimarisi.

2. Cihaz listesindeki her bir elemanın %80'ini (deney / testimize göre optimum dosya boyutu) kaplayan büyük boş bir dosya oluşturur. (Bu dosya her sürücü için sadece bir kez oluşturulmuştur). Bu açıklama Şekil 2.21'de tanımlanmıştır. Bilgisayardaki bir disk sürücüsüne karşılık gelen dosya, bir disk başlığına (disc header) ve 10 MB'lık boyuta sahip her bir bunchtaki dizi tabanlı bunch listesine sahiptir. Bunch listesinin son değeri boştur ve listenin bittiğin gösterir.

Çizelge 2.1. Kayıtlı IP listesi içeren sistem veri dosyası.

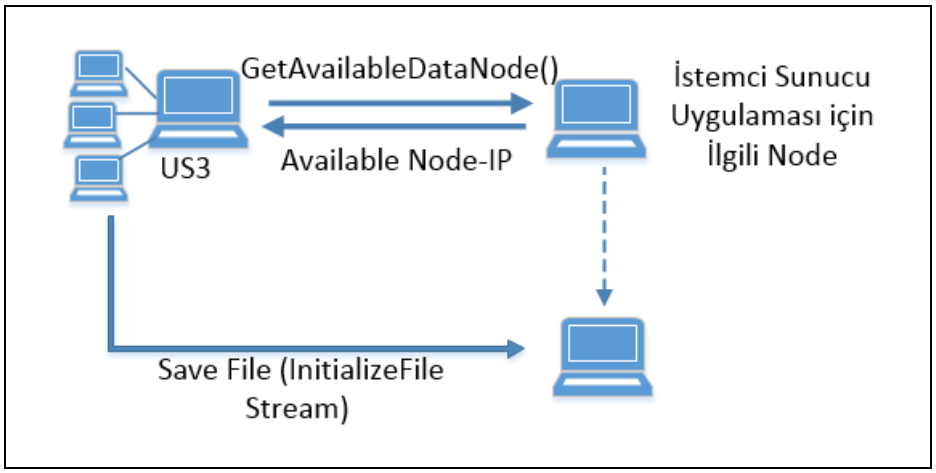
IP	Açıklama
xxx.xxx.xxx.xxx	Veri
xxx.xxx.xxx.xxx	Veri
xxx.xxx.xxx.xxx	Replica
xxx.xxx.xxx.xxx	Replica
xxx.xxx.xxx.xxx	İstemci Uygulama
xxx.xxx.xxx.xxx	İstemci Uygulama
xxx.xxx.xxx.xxx	İstemci Uygulama

3. Bu dosyanın yapısı Sanal Dosya Sistemi olarak aşağıda tanımlanmıştır.

Sistemin ilk node'unda bulunan SytemDataFile aktif olduğunda data node'u okur. Bu dosya sistemdeki tüm data node, replica node ve istemci uygulamalarının bilgilerini tutmaktadır. Bu, boyut olarak oldukça küçük bir büyüklüktedir. Çizelge 2.1'de gösterildiği gibi bütün data node'lar bilgisayar yeniden başlatıldığında bu listeyi IP değeri sabit olan data node'dan almaktadır. Daha sonra sisteme yeni bir ekleme olduğunda, sistem yöneticisi, servis yordam CheckServerList fonksiyonunu tüm data node'lara gönderir. Her bir istemci uygulaması kendisine ayrılan data noda bağlanır (en az bir adet data node). Kullanılan fonksiyonlar, GetAvailableDatanodeIp, SaveFile, ReadFile, DeleteFile, CheckServerList ve GetFileFromReplicadır.

2.2.6. SaveFile

İstemci uygulamasından sorumlu data node, yükleme dengeleyici (load balancer) gibi çalışır. Bu node saklanmak istenen verileri sıra ile diğer node'lara gönderir. Bu yüzden dosya kaydetme işlemi, iki aşamalıdır. İstemci uygulaması, dosya saklamak istediğinde sorumlu data node aracılığıyla uygun olan data node'un IP değerini alır. Daha önce bahsedildiği gibi, ilgili data node, istemci uygulamalarının, yönlendirme işleminden sorumlu olan bir proxy node'a sahip olduğu anlamına gelir.



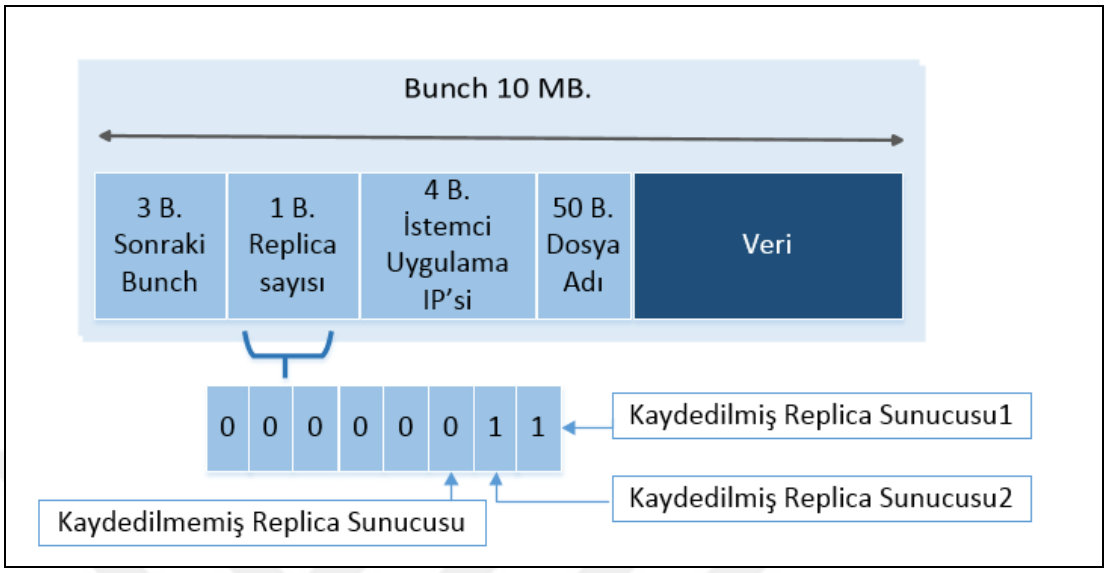
Şekil 2.22. Dosya Saklama Süreci

Şekil 2.22'de gösterildiği gibi, istemci dosyayı kaydetmeden önce proxy node'tan saklanacak olan dosyanın data node'nun IP adresini alır. Yük dengeleme işlevini, SaveFile işlemi ile farklı data node IP'si ile sağlar. Bu işlemi GetAvailableDatanodeIp fonksiyonu ile yapar. Uygun olan IP değerini alan istemci dosyasını bu IP de bulunan data node'a SaveFile fonksiyonu ile kaydeder. Data node, yazma işlemini her bir istekte yeni bir data node IP si vermek üzere yük dengeleme sağlamaktadır. SaveFile fonksiyonu dosyayı kendi dosya sistemine bağlı liste ve bitmap (hybrid of Unix bitmap ve Windows FAT32 yapısı revize edilmiş sürümü ile) yapısıyla kaydetmektedir. Bu işlem sonucunda client uygulamaya dosyanın hangi data node da olduğu ve başlangıç bucket değeri ile eşsiz dosya ismini geri döndürür. Özetle, istemci uygulaması dosya saklamak istediğinde, dosyasını saklayacağı data node'un IP adresini alır ve bu data node'a yazar. Bu sayede istemci hedefteki data node ile veri iletişimine geçer. Diğer sistemlerde, head node kullanan yapılarda, veri head node'a aktarılır ve head node bu veriyi data node'lara yazar. Burada transfer edilen veri miktarı iki kat artmaktadır. Çünkü istemciden head node'a ve buradan da data node'a gitmektedir. Bu tarz çalışma head node'un eşzamanlı bağlantı (concurrent connection) sayısını da hissedilir derecede düşürmektedir. Ancak bizim sistemimizde; istemci, dosyasını saklayacağı data node ile iletişime geçer, veri bir defa transfer edilir. Bu sayede eşzamanlı bağlantı sayısı da minimize edilmiş olur.

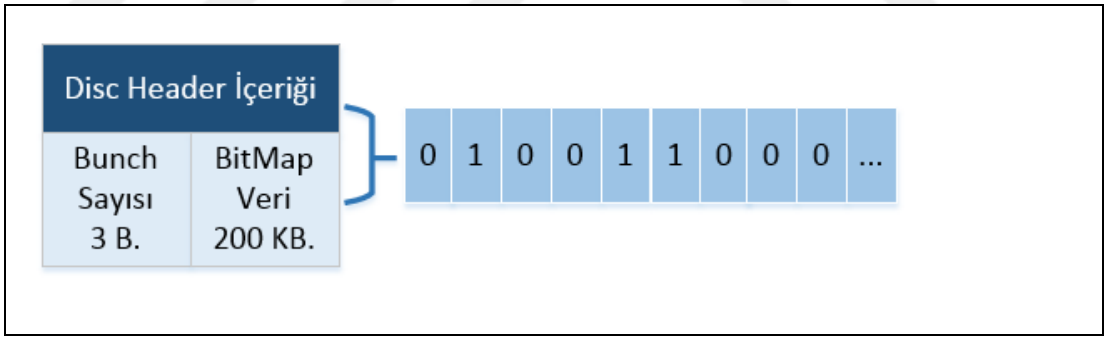
Bu değerleri alan istemci daha sonraki işlemlerinde (ReadFile ve DeleteFile komutlarında) sadece bu data node'a ulaşır ve isteklerini çalıştırır. Sadece ReadFile işlemi başarısız olduğunda istemci uygulaması GetFileFromReplica komutu ile dosyanın diğer kopyalarından birini alır.

Sisteme yeni bir istemci eklendiğinde ya da data node'larda bir değişiklik yapıldığında, sistem yöneticisi tarafından tüm node'lara CheckServerList komutu gönderilir.

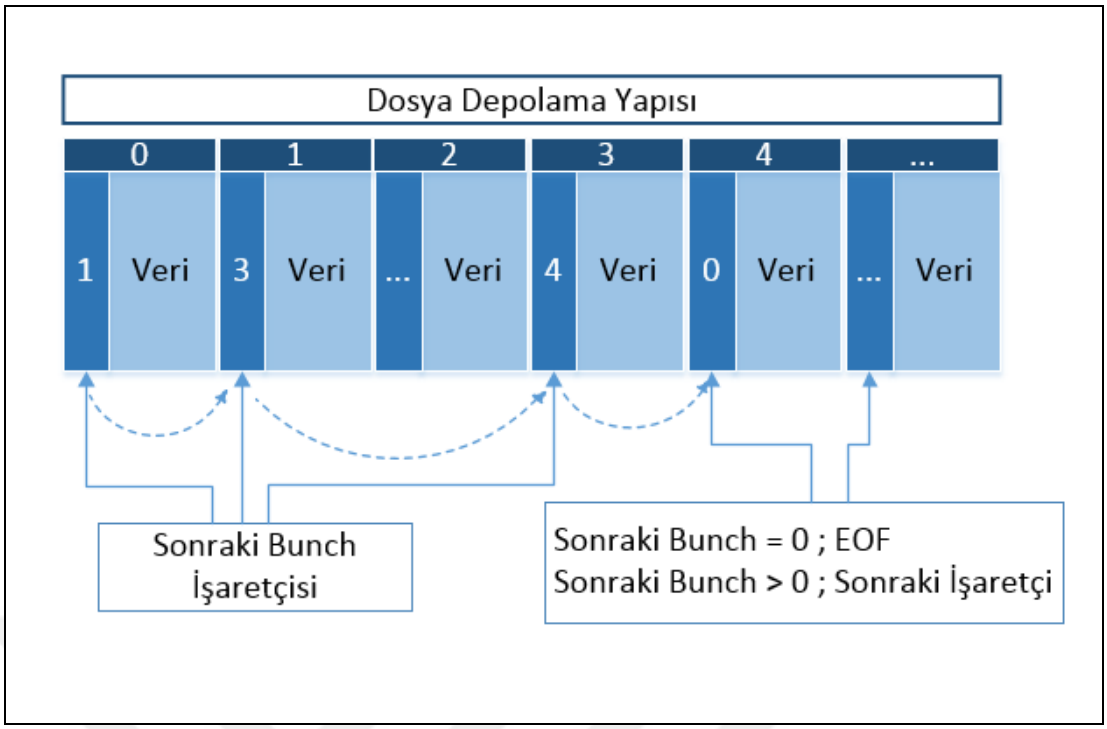
2.2.7. Virtual File System



Şekil 2.23. Dosya Sisteminin Bunch yapısı



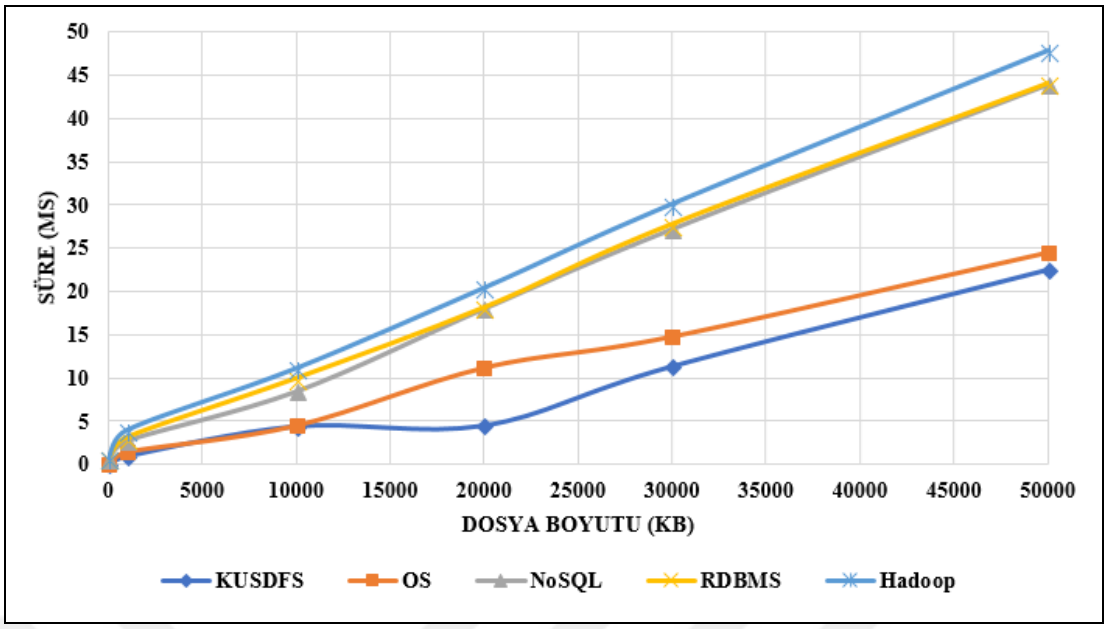
Şekil 2.24. Disc Header Yapısı



Şekil 2.25. Dosyanın Bunchlarının Bağlı-Liste Gösterimi

Çizelge 2.2. Farklı Dosya Büyüklüklerinde Sistemlerin Cevap Süreleri (ms.).

Dosya Boyutu (KB)	KÜADDS (ms.)	İşletim Sistemi (ms.)	NoSQL (ms.)	İVTYS (ms.)	Hadoop (ms.)
30	0,01	0,04	0,60	0,75	0,80
1000	0,92	1,43	2,74	3,15	4,01
10000	4,40	4,48	8,44	9,97	11,15
20000	4,48	11,19	18,01	18,16	20,45
30000	11,36	14,80	27,22	27,80	30,15
50000	22,60	24,54	43,95	44,08	47,88



Şekil 2.26. Farklı Dosya Büyüklüklerinde Sistemlerin Cevap Süreleri (ms.) Grafiği.

Sistem fiziksel aygıt üzerindeki toplam boş alanın %80'ini kaplayan bir dosya oluşturmaktadır. Bu durum Şekil 2.21'de gösterildiği gibi, dosya fiziksel sürücü üzerinde devam edecek şekilde bir alan oluşturacaktır. Böylece gerekli disk I/O'su azalacaktır. Bu dosya 10 MB lık bunch adı verilen bloklara bölünmektedir. Bu yapı Şekil 2.23'de gösterilmektedir. Bir bunch, 3 byte'lık bir sonraki bunch bilgisinden, 1 byte'lık veri işaretçisinden, 4 byte'lık istemci uygulaması IP'sinden, 50 byte'lık dosya adından ve veri blokundan oluşur. Bunch'lar, aynen bir dizi gibi davranır ve ilk index numarası 0'dan başlayarak devam eder. Şekil 2.24'de gösterildiği gibi, Disc Header bunch adedini tutan 3 byte'lık veriden ve bitmap formatında 200KB'lık veriden oluşur. Her bir bunch'ın boş olup olmadığı dosya yapısı içerisinde bulunan bitmap yapısı ile kontrol edilmektedir. Her bir bunch'ın bir bit ile temsil edildiği ve değeri 1 olanların kullanıldığı ya da ayrılmış bunch olduğu, boş olması durumunda, bunch'ın dosya zincirinde kullanım için hazır olduğunu durumunu gösterir. Bu bitmap alanı 200 KB tır. Bu sayede 16TB lık bir saklama alanına ulaşılmaktadır. Her bir bunch kendi içerisinde bir sonraki buch'ı tutan 4 B'lık işaretçiyi tutmaktadır. Bir buch'tan büyük olan dosyalar dosya zinciri ile tutulur. Şekil 2.25'de gösterildiği gibi,

Her bunch bir sonraki 3 byte'lık sonraki işaretçi ile sonraki veri bloğa işaret eder. İşaretçi sonraki bunch'un indexine sahiptir. Bu işaretçi 0 ise, bunch EndOfFile (EOF)-Dosya Sonu anlamına gelir. Başka bir deyişle, bu bunch dosya zincirinin sonuncusudur. Data Node'lar, kendilerine gönderilen dosyaları saklamak, okumak, silmek ve yedeklemekle sorumludur. Bu yapının en temel özelliklerinden biri de node'larda saklanan dosyanın başlangıç bunch numarası dosya adı ile gönderilir ve bu sayede okuma işlemi bir disk erişimi ile başlar. Bu özellik sayesinde sistem performansı oldukça tatmin edicidir. Bu durum Çizelge 2.2'de gösterilmiştir. KÜADDS ile diğer sistemlerin performans değerlerinin karşılaştırma sonuçları Şekil 2.26'daki grafikte gösterilmiştir.

Öncelikle geliştirilen sistem birden fazla istemci uygulama sunucusu tarafından kullanılabilmektedir. İstemci uygulama sunucusu, data node'lardan biri ile eşleştirilir ve tüm istekler bu sunucuya (data node) iletilir. Bir data node birden fazla istemci sunucusuna hizmet verebilir, bu sistemin dayananıklılık özelliğini arttırmaktadır. Uygulama suucuları farklı data node'lar ile iletişime geçebilir ve bu kendi sistemlerinin çalışmasını kesinlikle etkilemez. Sistemdeki data node'lardan her biri aynı anda birden fazla istemci sunucuya hizmet verebilir. Kısacası sistemin tüm node'ları istenildiğinde sunucu olarak kullanılabilmektedir. Bu özellik sayesinde çalışabilir durumda kalabilirlik (survivability), kullanılabilirlik (availability) ve güvenilirlik (reliability) değerleri oldukça iyi sonuç vermektedir.

İstemci uygulamaları kendisinin bağlı olduğu data node ile çalışır, istemci sunucusu dosya saklamak istediğinde data node'dan sıradaki data-node'un adresini ister. Bu adresi alan istemci sunucusu dosyayı bu IP adresindeki data node'a yazar. Böylece Uygulama sunucusu dosyalarını, sistemdeki data node'lara sırsıyla yazar. Uygulamanın ilişkilendirildiği data node'un tek görevi, kendisine bir IP adresi vermektir. Aslında bu yapı Hadoop mimarisindeki Name Node'a benzemektedir.

2.2.8. Replica Node

Dağıtık Dosya Sistemlerinde, sistemin hata toleransını arttırmak için bir yedekleme yöntemi olarak kullanılan replica node'ları, ayrıca bu sistemde yönetmek daha kolaydır. Öncelikle bu node'lar data node'ların sakladığı dosyaların kopyasını saklamak için kullanılır. Data node'lar kendisine yüklenen dosyaları asenkron olarak replica node'lara gönderir. Replica node'lar bu dosyaları temel işletim sistemi dosyası içerisinde saklar. Burada sadece işletim sisteminin dosyalama hizmeti kullanılır, özel bir dosya yapısı veya herhangi bir dosya işleme stratejisi kullanılmaz. Bu replica node'ların bir dosya sunucusu gibi çalıştığı anlamına gelir. Bu, replica node'ları, yalnızca işletim sistemi dizin yapısındaki dosyaları depolamaktan sorumlu olan dosya sunucusu gibi çalıştığı anlamına gelir. Replica sayısı, Dağıtık Dosya Sistemi için varsayılan çoğaltma sayısı olan üç adet olarak ayarlanmıştır ve bu değer tüm node'lar için sekize yükseltilebilecektir.

Bütün replica node'lar, sistemdeki dosyaları saklamak, silmek, güncellemek, iletmek ve aramaktan sorumlu bir servise sahiptir. Bu nedenle, bir dosya arama veya data node'lardan birisine veya istemciye ihtiyaç duyulduğunda, tüm replica node'larına SearchFile komutu gönderilir, daha sonra, aynı anda replica node'lar, hedeflenen dosyayı, kendi grid yapısındaki dizin yapılarında aramaya başlar, arama işleminin sonunda, her bir node, arama sürecinin başarılı olduğunu veya olmadığını gösteren arama sonuçlarını döndürür. Diğer bir deyişle, bir dosya bir node tarafından aranmak istendiğinde, tüm replica'lara SearchFile mesajları gönderilir, replica'lar paralel olarak çalışır ve sonuç, dosyayı isteyen data node'a iletilir. Replica node'ları yalnızca dosya aramaları yapmakla kalmaz, aynı zamanda bir dosyaya uygulanan işlemleri replica node'a yazma, okuma ve silme işlemlerini gösteren ReplicaWrite, ReplicaRead ve ReplicaDelete komutlarını da işlemektedir.

2.2.9. İşlevsel Özellikleri

1. Name node kullanımının yerine tercih edilen sunucusuz mimari: "Sunucusuz" terimi, sunucu kullanılmadığı veya sunucu olmadığı anlamına gelmez. Bu sadece sunucuya gerek kalmadığı anlamına gelmektedir [100]. Serverless aynı zamanda, hiyerarşik olarak aynı seviyedeki data node olarak ta adlandırılır. Bu çalışmada kullanılan 4 adet node'un her biri farklı uygulamalara hizmet vermektedir. Bu node'lardan herhangi biri devre dışı kaldığında sistem otomatik olarak diğer data node'lara yönlendirilecektir. Bu sayede;

   • "Sunucusuz" şema, sunucuların artık dahil olmadığı anlamına gelmez. Bu sadece geliştiricilerin artık onlar hakkında fazla düşünmemesi gerektiği anlamına gelir. Bilgi işlem kaynakları, fiziksel kapasiteler veya limitler etrafında çalışmak zorunda kalmaksızın hizmet olarak kullanılır.

   • Aynı makinede farklı kaynakları kullanmakta mümkündür.

2. Şirketler küçük ve orta büyüklükteki blok boyutlarına ihtiyaç duyarlar: Hadoop ve diğerleri genellikle en az 64 MB büyüklüğünde bir blok için konfigüre edilmişlerdir, bu yaklaşım küçük ve orta büyüklükteki büyük veriler için mükemmel bir seçim elde etmeyi zorlaştırır, bu durum şu şekilde sonuçlanır: küçük ve orta boy big data problemleri için iyi bir çözüm olamamaktadırlar.

3. Name node çökmesi sonucu ortaya çıkan sistem kurtarma problemi ortadan kaldırılmış oldu. Bu şekilde, sistemin hata toleransı çok yüksektir, çünkü name node yoktur.

4. Temel işletim sisteminde sanal dosya sistemi yönetimi kolay bir şekilde üretilmiştir. Sabit bir büyüklükte 10 MB olan bunch'ları yönetmek için hem bitmap hem de bağlantılı liste yapısının bir karışımı kullanılmıştır.

5. Dosyanın başlangıç adresi, yeterli performans artışı sağlayan istemci uygulamalarına kaydedilir.

6. Sistemimizin ortaya çıkarttığı önemli durumlardan bir tanesi, herhangi bir uygulama için herhangi bir data node'un sunucu olabileceğidir. Diğer bir deyişle, sistemdeki herhangi bir makine bir dosya sunucusu olabilir.

7. Data node içeren bütün IP listeleri ve replica IP'leri ve güvenli bağlantı IP'leri dizindeki bütün node'lar için erişilebilirdir.

KÜADDS'nin sahip olduğu fonksiyonel olmayan özellikler:

1. Performans (Reading Performance): Sistemin veriyi işleme hızını (özellikle kaydedilen verinin tekrar okunması) diğer sistemlerle karşılaştırdığımızda, iyi bir başarı sağlanmıştır.

2. Ölçeklenebilirlik (Scalability): Sahip olduğu özellikler sebebiyle yeni node eklemek için sadece servis programını yüklemek ve IP adresini diğer node'lara iletmek yeterlidir. Bu sayede sistem ihtiyaca göre birkaç node ile çalışabilirken binlerce node ile hizmet verebilecek şekilde tasarlanmıştır.

3. Çalışabilir durumda kalabilirlik (Survivability): Sistemin tüm olumsuzluklara rağmen hayatta kalması ve asgari fonksiyonlarını devam ettirmesidir. Geliştirilen sistemde node'lardan biri aktif olduğunda bile sistem çalışabilmektedir.

4. Kullanılabilirlik (Availability): Sistemin hizmetlerini sürekli başarılı bir şekilde sunmasıdır. Geliştirilen sistem ile bu gerçekleştirilmiştir. Özellikle hiç name node olmadığı için sistemin hayatta kalma direnci oldukça yüksektir.

5. Güvenlik (Security): Static IP ve simetrik şifreleme yöntemi: Saklanan veriler her istemci kendi verilerini özel anahtar ile depolamaktan sorumlu olacak şekilde tamamen istemci uygulama hedefleri için tasarlanmıştır.

6. Minumum maliyet: Sistemde kullanılan data node'lar sıradan makineler olup ilave bir özelliğe sahip değildir. Böylece düşük maliyetli bir yapı geliştirilmiştir.

3. SONUÇLAR VE TARTIŞMA

3.1. Sonuçlar

Bu çalışmada, özellikle küçük ve orta büyüklükteki big data setleri için hızlı, güvenli, sunucusuz, güçlü, sağlam, yönetilebilir, ölçeklendirilebilir dağıtık dosya sistemi geliştirildi. İlk önce big data'yı oluşturacak EBYS sistemi tasarlandı. Kırıkkale Üniversitesi için tasarlanmış akıllı dağıtık dosya sistemi (KÜADDS) geliştirildi. Sistem, çoğu dağıtık sistemlerin karşısında gelişmiş bir sistem platformudur. TCP/IP protokolünü kullanır. Server node, head node veya name node kullanılmamaktadır. Sistem sunucusuzdur. Bu sayede sistemin sürekliliği sağlanmış olur. Bir data node sistemde düzgün çalışmadığında, diğer node'lar istekleri cevaplayabilir ve sistemin çalışması devam eder. Gerekirse, sadece windows servis yordamı node'una yükleyerek sisteme sınırsız sayıda data node'lar kolaylıkla eklenebilir. Bu sistemin diğer sistemlere göre üstünlüğüdür.

Sistem güvenliği, diğer dağıtık dosya sistemlerinde olduğu gibi kabul edilebilir bir düzeye sahiptir. Güvenlik iki yolla sağlanmaktadır. Birincisi, data node'tan hizmet veren istemci makinelerin IP'ler kontrol edilerek, ikincisi ise, uygulama yazılımının data node'lara gönderdiği verileri şifrelemektir.

Bir data node sistemimizde birden fazla istemciye hizmet verebileceğinden, diğer dağıtık dosya sistemlerinden daha iyi bir yük dengeleme performansına sahiptir. Aynı şekilde, bir istemci birden fazla data node'a veri yükleyebilir.

Tasarlanan sistemde, bir disk, "bunch" ve "disc header" olarak adlandırılan data setlerden oluşmaktadır. Disc header, diskte bulunan bunch sayısını ve bunch'un bitmap yapısıyla boş olup olmadığını tutar. Bir bunch, bir sonraki bunch'ı, verilerin replicalarını, verileri yükleyen istemci uygulamasının IP'sini ve verilerin kendisini tutar. Tasarlanan dosya sistemini kullanarak, her bunch dosya zincirinin bir üyesi olan bir sonraki bunch'ı işaret eder.

Replica işlemi, işletim sisteminin dosya işlemleri sistem çağrılarını kullanır (Windows'da API'ler). Data node'lar, kendilerine eş zamansız olarak yüklenen dosyaları replica node'lara gönderir.

Tasarlanan dağıtık dosya sistemi diğer dosyalama sistemleri ile karşılaştırılmıştır. Analizimize göre, sistemimiz NoSQL sisteminden %97, İlişkisel Veri Tabanı Yönetim Sisteminden %80, işletim sisteminden %74 daha iyi performans göstermiştir.

Gelecekte, EBYS kullanan kuruluşlar KÜADDS'e entegre edilebilir. Bu sayede, kurumların maliyeti, iş yükü ve teknoloji ihtiyaçları en aza indirilecektir. Kendi arşivleme sistemlerini kurmak yerine, sadece kütüphane dosyalarını yükleyerek sisteme erişmek ve sisteme entegre etmek mümkündür. Geliştirilen sistem, kurumlar için uzmanlaşmış 10 MB'lık sabit bir bunch boyutuna sahiptir. Farklı dosya boyutlarına ve türlerine sahip olan farklı uygulamaları kapsaması için, gelecekteki çalışmalarda dinamik bir bunch boyutu geliştirilebilir. Sistemi, hesaplama hızları açısından daha güçlü ve verimli hale getirmek için grid hesaplama işlevleri de uygulanabilir. Bu gelecekteki araştırmalar için önemli bir konudur. Ek olarak, son yıllarda daha yaygın olarak kullanılan GPU'lar, daha spesifik bir dağıtık bilgi işlem platformu elde etmek ve grid performansını arttırmak için sisteme dahil edilebilir. Ancak, grafik işlemci biriminin gücünü kullanarak hesaplama performansının artmasını sağlayan bir sistemin tasarımını yapmak için odaklanmak gerekmektedir.

3.2. Tartışma

Dünya üzerinde üretilen büyük veriyi yönetmek ve depolamak için yapılan çalışmalar son yıllarda artmıştır. Özellikle Hadoop veonun ekosistemini kullanan çalışmalar son derece fazla kullanılmaktadır. Ancak, yapılan çalışmalara bakıldığında bunların çoğunlukla büyük veri analizi çalışmaları olduğu görülmektedir. Aynı şekilde bu çalışmalar MapReduce işlevlerinin veriler üzerinde

nasıl kullanıldığına dair tanımlayıcı uygulamalar olmuştur. Ancak, küçük, orta ve büyük kurumların ihtiyaçları için minimum iç paylaşım ile daha verimli veri depolama işlemi için ne yapılacağını ele almamışlardır. Bu çalışma, bu eksikleri giderecek olan çözümleri ele almış ve bunlara cevap bulmaya yardımcı olacak başarılı bir çalışma olmuştur.

Bu araştırmanın, big data problemini çözmek için tek başına kullanılabilecek bir sistem olması düşünülmüş ve uygulanmıştır.

Bu çalışmanın güçlü yönleri şunlardır:

1. Hybrid mimariden dolayı okuma yazma performansı iyidir.

2. Name node'un kullanılmamış olması ve her node'un bir sunucu gibi kullanılması sayesinde sağlamlık ve kullanılabilirlik sağlanmıştır.

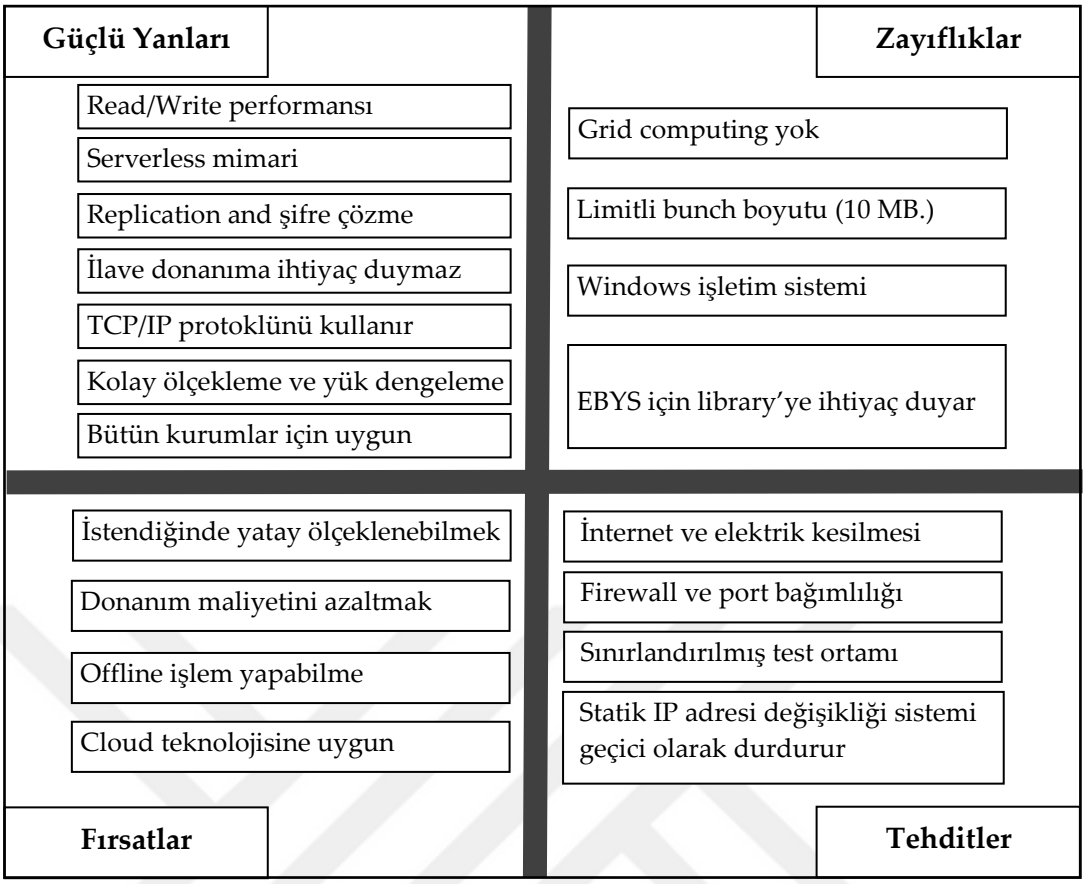
3. En iyi veri yük dengeleme sağlanmıştır.

4. Ucuz, standart, var olan donanım elemanlarını dahi sisteme entegre edilebiliyor olması.

5. 128-bit simetrik kriptoloji kullanılarak bağlantı güvenli hale getirilmiştir.

6. Kütüphane dosyalarını node'lara kolayca yükleyerek, kolay ölçeklenebilirlik sağlanmıştır.

7. 10 MB blok büyüklüğünde kurumlar için en uygundur. Şekil 3.1, sistemin güçlü, zayıf yönlerini ve fırsat özelliklerini göstermektedir.



Şekil 3.1. Amaçlanan sistemin SWOT analizi

Bu çalışmanın kısıtları da aşağıda listelenmiştir.

1. Grid hesaplama kolaylıkları yoktur.

2. 10 MB sabit bunch boyutunun olması.

3. Geçici sistemin durmasına neden olabilecek olan statik IP adresi değişikliği, ancak bu durum yalnızca istemci uygulamasında geçerli olacaktır, dolayısıyla sistemin kullanılabilirliğini etkilemez.

4. Test ortamının dört node ile sınırlı olması.

5. Simetrik kriptografi tek bir anahtarla çalışır ve bu şifrenin güvenliği için anahtarın gizli kalması önemlidir.

Burada açıklananların projeksiyonu doğrultusunda, bu çalışmanın avantajları ve dezavantajları bulunmaktadır. Bu sistemin dezavantajlarını geliştirmek için gelecekteki araştırmaların yapılması önemlidir.

KAYNAKLAR

[1] Anonim, Internet Growth Statistics, https://www.internet worldstats.com/emarketing.htm (Erişim Tarihi:14.05.2018)

[2] Anonim, Total number of Websites, http://www.internetlivestats.com/total-number-of-websites/ (Erişim Tarihi:30.05.2018)

[3] R. H. Zakon, Hobbes' Internet Timeline 25, https://www.zakon.org /robert/internet/timeline/ (Erişim Tarihi:21.04.2018)

[4] R. Jacobson, 2.5 quintillion bytes of data created every day. How does CPG & Retail manage it?, https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it/ (Erişim Tarihi:15.04.2018)

[5] B. Walker, Every Day Big Data Statistics – 2.5 Quintillion Bytes Of Data Created Daily, http://www.vcloudnews.com/every-day-big-data-statistics-2-5-quintillion-bytes-of-data-created-daily/ (Erişim Tarihi:12.03.2018)

[6] L. Mearian, Scientists calculate total data stored to date: 295+ exabytes, https://www.computerworld.com/article/2513110/data-center/scientists-calculate-total-data-stored-to-date--295--exabytes.html (Erişim Tarihi: 30.03.2018)

[7] C. Reuter, A. L. Hughes, M. A. Kaufhold, Social Media in Crisis Management: An Evaluation and Analysis of Crisis Informatics Research, International Journal of Human-Computer Interaction, January 2018, DOI: 10.1080/10447318.2018.1427832.

[8] Anonim, https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/ (Erişim Tarihi:22.04.2018)

[9] P. Dhavalchandra, M. Jignasu and R. Amit, Big Data - A Survey of Big Data Technologies, International Journal of Science Research and Technology, vol. 2, no. 1, pp. 45-50, 2016.

[10] B. B. Dean, Use of Electronic Medical Records for Health Outcomes Research, Medical Care Research and Review, vol. 66, no. 6, pp. 611-638, 2009.

[11] A. Erguzen ve E. Erdal, "Medical Image Archiving System Implementation with Lossless Region of Interest and Optical Character Recognition," Journal of Medical Imaging and Health Informatics, vol. 7, pp. 1-7, 2017.

[12] I. D. Dinov, "Volume and Value of Big Healthcare Data," Journal of Medical Statistics and Informatics, 4: 1-7, 2016.

[13] M. Chen, S. Mao, Y. Liu, Big Data: A Survey, Mobile Network Application, 19 (2): 171–209, 2014

[14] Anonim, China National Genebank Officially Opens, https://www.bgi.com/global/company/careers/china-national-genebank-officially-opens/ (Erişim Tarihi:20.04.2018)

[15] J. E., Johnson., Big Data + Big Analytics = Big Opportunity, Financial Executive, 28(6): 50-53, 2012.

[16] Anonim, Industry Trends, https://www.gminsights.com/industry-analysis/electronic-document-management-system-market, (Erişim Tarihi: 15.05.2018)

[17] Anonim, Document Management Systems Market, https://www.marketsandmarkets.com/Market-Reports/document-

management-system-market-229353320.html?gclid=Cj0KCQjwjN7Y BRCOARIsAFCb934hicurierj9z_VHEHtodigQTzBsdFnXkSvmqxqbiS5pEve aRiGTxoaAh3sEALw_wcB, (Erişim Tarihi: 30.05.2018)

[18] N. Elgendy and A. Elragal, Big Data Analytics: A Literature Review Paper, 14th Industrial Conference, ICDM 2014, St. Petersburg, Russia, 2014.

[19] N. Gürsakal, Büyük Veri, Dora Yayıncılık, 2. Baskı, Bursa, Türkiye, 2014.

[20] P. A. Alsberg and J. D. Day, A Principle for Resilient Sharing of Distributed Resources, ICSE '76 Proceedings of the 2nd international conference on Software engineering, San Francisco, California, USA, 1976.

[21] C. A. Ellis and R. A. Floyd, The ROE File System, 3rd Symposium on Reliability in Distributed Software and Database Systems, Clearwater Beach, FL, USA, 1983.

[22] G. Popek, B. Walker, J. Chow and D. Edwards, LOCUS a network transparent, high reliability distributed system, SOSP '81 Proceedings of the eighth ACM symposium on Operating systems principles, Pacific Grove, California, USA, 1981.

[23] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh and B. Lyon, Design and Implemantation of The Sun Network File System, USENIX Conference and Exhibition, Portland, Oregon, USA, 1985.

[24] G. Coulouris, J. Dollimore, T. Kindberg and G. Blair, Distributed Systems: Concepts and Design 5th, USA: Addison-Wesley Publishing Company, 2011.

[25] S. Heidl, Evaluierung von AFS/OpenAFS als Clusterdateisystem, Berlin: Tech. rep., Zuse-Institut Berlin, http://www2.cs.upb.de/StaffWeb/jens/

Courses/VLZIB/Datamngmnt/ausarbeitung_sebastian_afs.pdf, 2001. (Erişim Tarihi: 18.12.2008).

[26] P. Bzoch, J. Šafarik, Algorithms for Increasing Performance in Distributed File Systems, Acta Electrotechnica et Informatica, 12 (2): 24-30, 2012.

[27] B. Karasulu, S. Korukoğlu, Modern Dağıtık Dosya Sistemlerinin Yapısal Karşılaştırılması, Akademik Bilişim'2008, Çanakkale, Türkiye, 2008.

[28]. C. A. Thekkath, T. Mann, E. K. Lee, Frangipani: a scalable distributed file system, SOSP '97 Proceedings of the sixteenth ACM symposium on Operating systems principles, Saint Malo, France, 1997.

[29] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. P. Wattenhofe, FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment, USENIX 5th Symposium on Operating Systems, Boston, Massachusetts, USA, 2002.

[30] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long and C. Maltzahn, Ceph: A Scalable, High-Performance Distributed File System, OSDI '06 7th Symposium on Operating Systems Design and Implementation, Seattle, Washington, USA, 2006.

[31] K. Shvachko, H. Kuang, S. Radia, The Hadoop Distributed File System, Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, Incline Village, NV, USA, 2010.

[32] G. Yavuz, S. Aytekin ve M. Akçay, Apache Hadoop Ve Dağıtık Sistemler Üzerindeki Rolü, Dumlupınar Üniversitesi-Fen Bilimleri Enstitüsü Dergisi, 27: 43-54, 2012.

[33] Anonim, https://en.wikipedia.org/wiki/Apache_Hadoop. (Erişim Tarihi: 27.06.2016).

[34] A. Thomson, D. J. Abadi, CalvinFS: Consistent WAN Replication and Scalable Metadata Management for Distributed File Systems, 13th USENIX Conference on File and Storage Technologies (FAST ’15), Santa Clara, CA, USA, 2015.

[35] M. S. Al-Kahtani, L. Karim, An Efficient Distributed Algorithm for Big Data Processing, Arabian Journal for Science and Engineering, 42 (8): 3149-3157, 2017.

[36] Q. Wang, H. Wang, C. Zhang, W. Wang, Z. Chen, F. Xu, A parallel implementation of idea graph to extract rare chances from big data, IEEE International Conference on Data Mining Workshop, Shenzhen, 2014.

[37] Z. Liang, W. Li and Y. Li, A parallel probabilistic latent semantic analysis method on MapReduce platform, 2013 IEEE International Conference on Information and Automation (ICIA), Yinchuan, 2013.

[38] T. Chen, H. Wei, M. Wei, Y. Chen, T. Hsu and W. Shih, LaSA: A locality-aware scheduling algorithm for Hadoop- MapReduce resource assignment, 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, USA, 2013.

[39] M. Muja, D. Lowe, Scalable Nearest Neighbor Algorithms For High Dimensional Data, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (11): 2227 - 2240, 2014.

[40] F. Lu, L. Hong and L. Changfeng, The improvement and implementation of distributed item-based collaborative filtering algorithm on Hadoop, 2015, 34th Chinese Control Conference (CCC), Hangzhou, Chine, 2015.

[41] V. Cevher, S. Becker, M. Schmidt, Convex Optimization For Big Data: Scalable, Randomized, And Parallel Algorithms For Big Data Analytics, IEEE Signal Process. Mag., 31 (5): 32-43, 2014.

[42] M.S. Hossaina, G. Muhammad, Cloud-Assisted Industrial Internet of Things (IIoT)-Enabled, Computer Networks, 101: 192-202, 2016

[43] T. Ryan, Y.C. Lee, Multi-Tier Resource Allocation for Data-Intensive Computing, Big Data Research, 2(3): 110-116, 2015.

[44] M. A. Martínez-Prieto, C. E. Cuesta, M. Arias, J. D. Fernández, The Solid Architecture For Real-Time Management of Big Semantic Data, Future Generation Computer Systems, 47, 62-79, 2014.

[45] J. Maccormick, N. Murphy, M. Najork, C. A. Thekkath, L. B. Zhou, Boxwood: Abstractions As The Foundation For Storage Infrastructure, OSDI '04: 6th Symposium on Operating Systems Design and Implementation, San Fransisco, 105-120, 2004

[46] A. K. Sahoo, K. S. Sahoo, M. Tiwari, GPU based Parsing and Searching of XML Document for Multimedia data using MapReduce Approach , IEEE CALCON, Kolkata, 2015.

[47] Anonim, The Basics of Document Sizing, http://www.supportingadvancement.com/records/document_imaging_sizing/document_imaging_sizing.htm, (Erişim Tarihi: 15.05.2018).

[48] Anonim, Earth Observing Laboratory, https://www.eol.ucar.edu/field_projects/bomex, (Erişim Tarihi: 10.05.2018).

[49] G. Halevi, H. F. Moed, The Evolution of Big Data as a Research and Scientific Topic: Overview of the Literature, Research Trends, 30, 1-7, 2012.

[50] E. Yetimler, Internet of Things (Nesnelerin İnterneti) Nedir? Cihazların Etkileşim Trendleri, https://www.karel.com.tr/blog/internet-things-nesnelerin-interneti-nedir-cihazlarin-etkilesim-trendleri, (Erişim Tarihi: 12.05.018).

[51] Anonim, Web Of Science Arama sonucu, http://0211h0tnz.y.http.apps.webofknowledge.com.proxy.kirikkale-elibrary.com/RAMore.do?product=WOS&search_mode=GeneralSearch&SID=D5dXYSTPT5zhqIveXUb&qid=2&ra_mode=more&ra_name=PublicationYear&colName=WOS&viewType=raMore, (Erişim Tarihi: 30.05.2018).

[52] Anonim, Google Trends, https://trends.google.com/trends/explore?date=all&q=big%20data, (Erişim Tarihi: 25.05.2018).

[53] A. Gandomi, M. Haider, Beyond the Hype: Big Data Concepts, Methods, And Analytics, International Journal of Information Management, 35 (2): 137-144, 2015.

[54] D. Laney, 3D Data Management Controlling Data: Volume, Velocity and Variety, https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf, 2011, (Erişim Tarihi: 28.05.2018).

[55] H. Chen, R. H. L. Chiang, V. C. Storey, Business Intelligence and Analytics: From Big Data to Big Impact, Business Intelligence Research, 36 (4): 1165-1188, 2012.

[56] Anonim, Gartner IT Glossary, http://www.gartner.com/it-glossary/big-data/, (Erişim Tarihi: 15.05.2018).

[57] S. Mills, S. Lucas, Demystifying Big Data: A Practical Guide To Transforming The Business of Government, TechAmerica Foundation's Federal Big Data Commission, 2012.

[58] M. Bărbulescu, R. Grigoriu, Integrating of Structured, Semi-structured and Unstructured Data in Natural and Build Environmental Engineering, 2013 11th RoEduNet International Conference, 17-19 Jan. 2013, Sinaia, Romania, 2013.

[59] L. Belbin, M.P. Austin, C. R. Margules., I. D. Cresswell, R. Tharckway, Data Suitability, Modelling of Landscape Patterns and Processes Using Biological Data, 1995.

[60] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, P. Tufano, Analytics: The Real-World Use of Big Data. How Innovative Enterprises Extract Value From Uncertain Data. IBM Institute for Business Value, 2012.

[61] C. K. Emani, N. Cullot, C. Nicolle, Understandable Big Data: A Survey, Computer Science Review, 17, 70-81, 2015

[62] K. Cukier, The Economist, Data, data everywhere: A special report, http://www.economist.com/node/15557443, 2015, (Erişim Tarihi: 25.05.2018).

[63] Anonim, Big Data Now: 2014 Edition, O'Reilly Media, https://www.oreilly.com/ideas/big-data-now-2014-edition, (Erişim Tarihi: 22.05.2018).

[64] S. Alkan, Veri Dünyasının Geleceği : Büyük Veri (Big Data), Garanti Teknoloji, https://tr.linkedin.com/pulse/veri-d%C3%BCnyas%C4%B1n%C4%B1n-gelece%C4%9Fi-b%C3%BCy%C3%BCk-big-data-seydi-alkan (Erişim Tarihi: 25.05.2018)

[65] X. L. Dong, D. Srivastava, Big Data Integration, Morgan & Claypool Publishers, Williston, 2015

[66] K. U. Jaseena, J. M. David , Issues, Challenges, And Solutions: Big Data Mining, Sixth International Conference on Networks & Communications, Vienna, Austria, 2018.

[67] M. Milli, F. Şentürk, S. Çınaroğlu, İ. Çınaroğlu, Büyük Veri Kavramı ve Karakteristik Özellikleri, Akademik Bilişim'2016, Aydın, Türkiye.

[68] A. Ergüzen, M. Ünver, Developing a File System Structure to Solve Healthy Big Data Storage and Archiving Problems Using a Distributed File System, Applied Science, 8(6): 913, 2018.

[69] R. Cattell, "Scalable SQL and NoSQL Data Stores", ACM SIGMOD Record, 39:12- 27, 2011.

[70] Kris Zyp, NoSQL Architecture, https://www.sitepen.com/blog/ 2010/05/11/nosql-architecture/, (Erişim Tarihi: 26.06.2018).

[71] A. Nayak, A. Poriya, D. Poojary, Type of NOSQL Databases and Its Comparison With Relational Databases, International Journal of Applied Information Systems, 5(4): 16-19, 2013

[72] U. Aslan, Hangi NoSQL Veritabanı?, https://kodcu.com/2016/03/hangi-nosql-veritabani/, (Erişim Tarihi: 24.06.2018).

[73] J. Klein, I. Gorton, N. Ernst, P. Donohoe, K. Pham and C. Matser, Application-Specific Evaluation of NoSQL Databases, 2015 IEEE International Congress on Big Data (BigData Congress), New York City, NY, USA, 2015.

[74] S. Davaz, NoSQL Nedir Avantajları ve Dezavantajları Hakkında Bilgi, https://blog.kodcu.com/2014/03/n-osql-nedir-avantajlari-ve-dezavantajlari-hakkinda-bilgi/ (Erişim Tarihi:13.06.2017).

[75] A. Leslie, What is Hadoop Good For? (Best Uses, Alternatives, & Tools), https://www.hostingadvice.com/how-to/what-is-hadoop/, (Erişim Tarihi: 28.06.2018)

[76] A. Silberschatz, P. B. Galvin,Greg Gagne, "Operating System Concepts 7th Ed", John Wiley&Sons,611,(2004)

[77] G. Yavuz, S. Aytekin, M. Akçay, Apache Hadoop ve Dağıtık Sistemler Üzerindeki Rolü, Dumlupınar Üniversitesi-Fen Bilimleri Enstitüsü Dergisi, 27, 43-54, 2012.

[78] K. Shvachko, K. Hairong, S. Radia,R. Chansle, "The Hadoop Distributed File System", Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium, Incline Village, NV, USA (2010)

[79] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, (2006)

[80] H. He, Z. Du, W. Zhang, A. Chen, Optimization Strategy of Hadoop Small File Storage For Big Data In Healthcare, The Journal of Supercomputing, 72 (10): 3696–3707, 2016.

[81] W. Raghupathi, V. Raghupathi, Big Data Analytics In Healthcare: Promise And Potential, Health Information Science And Systems, 2(1):1-6, 2014.

[82] Ş. E. Şeker, İş Zekâsı ve Veri Madenciliği, İstanbul, Cinius, 2013.

[83] M. Yılmaz, Enformasyon ve Bilgi Kavramları Bağlamında Enformasyon, Yönetim ve Bilgi Yönetimi.” Ankara Üniversitesi Dil ve Tarih-Coğrafya Fakültesi Dergisi, 49(1): 95-118, 2009

[84] K. Jeffery, Data is the New Oil, Best Practices for Data Management & Sharing. Dü. The Joint Research Centre (JRC), Ispra, Italy, 2014, (Erişim Tarihi: 13.03.016

[85] K. Doğan, S. Arslantekin, Büyük Veri: Önemi, Yapisi Ve Günümüzdeki Durum, DTCF Dergisi,56(1): 15-36, 2016.

[85] Anonim, Veri ve Bilgi Arasındaki Fark, https://limenya.com/veri-ve-bilgi-arasindaki-fark/, (Erişim Tarihi: 25.06.2018)

[87] ISO (2001). ISO 15489-1: Information and Documentation – Records Management – Part 1: General. http://www.iso.org, (Erişim tarihi: 09.10.2009)

[88] Resmî Gazete, 29255, 02.02.2015. http://www.resmigazete.gov.tr/eskiler/2015/02/20150202-1.htm, (ErişimTarihi:25.06.2018)

[89] M. Bektaş, Elektronik Belge Yönetim Sistemi (EBYS)’nin İnsan Kaynaklarinin Dönüşümüne Etkisi: Marmara Üniversitesi Örneği, Doktora Tezi, Marmara Üniversitesi, İstanbul, 2015.

[90] E. Esgin, Kamuda Kurumsal Bilgi Yönetimi İçin e-dönüşüm Modeli: Marmara Üniversitesi Elektronik Belge Yönetim Sistemi Örneği, Marmara Üniversitesi, Doktora Tezi, İstanbul, 2015

[91] M. Cook, Information Management and Archival Data, London: The Library Association, 1993

[92] E. Shepherd, Managing Electronic Records, Records Management Journal, 4(1), 39-49, 1994.

[93] C. Aydın, F. Özdemirci, Elektronik Belgelerin Arşivlenmesinde Gerçekliğin ve Bütünlüğün Korunması, Bilgi Dünyası, 12 (1): 105-127, 2011.

[94] Ö. Külcü, Değişen Koşullarda Belge Yönetimi Çalışmaları ve Uluslararası Uygulamalar, XII. Türkiye'de İnternet Konferansı, 8-10 Kasım: 64, Ankara, Türkiye.

[95] L. Fernandez, J. T. Sprehe, Integrating an ERDMS in an IT Environment, Information Management Journal; 37 (4): 58, 2003.

[96] T.C: Başbakanlık, 2008/16 sayılı Genelge. http://www.resmigazete. gov.tr/eskiler/2008/07/20080716-7.htm, (Erişim Tarihi: 26.06.2018)

[97] Ş. Eroğlu, Ö. Külcü, TS 13298 Çerçevesinde Kurumsal Bilgi Sistemleri ve Elektronik Belge Yönetimi Standartlarının Değerlendirilmesi: İçişleri Bakanlığı Örneği, Bilgi Dünyası, 15 (2): 327-352, 2014

[98] M. B. K. Önaçan, T. Medeni, D. Ö. Özkanlı, EBYS'nin Faydaları ve Kurum Bünyesinde EBYS Yapılandırmaya Yönelik Bir Yol Haritası", Sayıştay Dergisi, 85: 1-26 , 2012

[99] Anonim, Yalova Çevre ve Şehircilik İl Müdürlüğü, http://yalova.csb.gov.tr/mudurlugumuz-2010-2017-yillari-arasi-evrak-hareketliligi-haber-221643, http://www.emo.org.tr/ekler/b79bbad9d843006_ek.pdf?tipi=20&turu=H&sube=7 http://www.ktu.edu.tr/dosyalar/iibf_89483.pdf, (Erişim Tarihi: 25.05.2018).

[100] K. Fromm, Readwrite, http://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless/ (Erişim Tarihi: 15.06.2017).

EKLER

EK1. login.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="login.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Kırıkkale Üniversitesi - EBYS</title>

   <style type="text/css">
       body {
     background-image: url(images/arkaplan-mavi2.jpg);
}
       .container {
    display:block;
    text-align: center;
    vertical-align: middle;
}
#login-box {
     width:350px;
     height: 450px;
     padding: 48px 76px 0 60px;
     color: #ebebeb;
     font: 12px Arial, Helvetica, sans-serif;
     background-image:url(images/login-box-backg.png);
    background-repeat:no-repeat;
    background-position:left;
    background-position-y:top;
    display: inline-block;

}
#login-box img {
     border:none;
}
#login-box h2 {

     padding:0;
     margin:0;
     color: #ebebeb;
     font: bold 40px "Calibri", Arial;
}
#login-box-name {
     float: left;
```

```
     width:80px;
     text-align: right;
     padding: 14px 10px 0 0;
     margin:0 0 7px 0;
}
#login-box-name2 {
     float: left;
    width:80px;
     text-align: right;
     padding: 14px 10px 0 0;
     margin:0 0 7px 0;
}
#login-box-field {
     float: left;
     display:block;
     width:230px;
     margin:0;
     margin:0 0 7px 0;
}
#login-box-field2 {
     float: left;
     display:block;
     width:230px;
     margin:0;
     margin:0 0 7px 0;
}
.form-login  {
     width: 205px;
     padding: 10px 4px 6px 3px;
     border: 1px solid #0d2c52;
     background-color:#1e4f8a;
     font-size: 16px;
     color: #ebebeb;
}
.login-box-options  {
     clear:both;
    display:block;
     padding-left:87px;
     font-size: 11px;
}
.login-box-options a {
     color: #ebebeb;
     font-size: 11px;
}
       .btn {
           margin-left:0px;
       }
       #Div1 {
```

```
            width: 86px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
     <div  class="container">
         <div id="login-box">
             <h2> <dx:ASPxImage ID="ASPxImage2"
runat="server" ShowLoadingImage="true" Height="60px"
ImageUrl="~/images/KkuLogo3.png"
Width="60px"></dx:ASPxImage>     EBYS-
Giriş</h2>

             <br />
             <span style="font-weight:bold">  Kırıkkale
Üniversitesi - Elektronik Belge Yönetim Sistemi</span><br
/>
             <br />
             <div id="login-box-name" style="margin-top:
20px;">Kullanıcı Adı:</div>
             <div id="login-box-field" style="margin-top:
20px;">
                 <asp:TextBox ID="TextBox1"
runat="server" Cssclass="form-login"
MaxLength="2048"></asp:TextBox>
             </div>
             <div id="login-box-name2">Şifre:</div>
             <div id="login-box-field2">
                 <asp:TextBox ID="TextBox2"
runat="server" Cssclass="form-login"
MaxLength="2048"></asp:TextBox></div>
             <br />
             <span class="login-box-options">
                 <asp:CheckBox ID="CheckBox1"
runat="server" />
                 Beni Hatırla<a href="#" style="margin-
left: 30px;">Şifremi Unuttum?</a></span>
             <dx:ASPxButton ID="ASPxButton1"
runat="server" Text="" BackColor="Transparent"
Width="103" Height="42" CssClass="btn"
Theme="MetropolisBlue" OnClick="ASPxButton1_Click1">
                 <Image Url="~/images/login-btn.png">
                 </Image>
                 <Border BorderStyle="None" />
             </dx:ASPxButton>
         </div>
        </div>
```

```
    </form>
</body>
</html>
```

EK2. login.aspx.cs

```
    protected void ASPxButton1_Click1(object sender,
EventArgs e)
    {
        SqlConnection scon = new
SqlConnection(sabitler.constring);
        string komut = @"select br.id as
birimid,br.anabirim,s.unvani,br.birimadi,s.id,s.adi,order
no,s.soyadi,u.* from users u, sahislar s,birimler br
where s.id=u.ownersahis and br.id=s.ownerbirim and
u.username='" + TextBox1.Text + "' and u.password='" +
TextBox2.Text + "' ";
        scon.Open();
        SqlCommand scom = new SqlCommand(komut, scon);
        SqlDataReader dr = scom.ExecuteReader();
        if (dr.HasRows)
        {
            dr.Read();
            Session["birimid"] = dr["birimid"];
            //  Session["gelenevrakid"] = "1019";
            Session["anabirim"] = dr["anabirim"];
            Session["gidenevrakid"] = "1019";
            Session["kullaniciid"] = dr["id"];
            Session["birimadi"] = dr["birimadi"];
            Session["unvani"] = dr["unvani"];
            Session["adi"] = dr["adi"];
            Session["soyadi"] = dr["soyadi"];
            Session["orderno"] = dr["orderno"]; // birim
yetki sınıfı

            Session["ustyonetici"] = dr["ustyonetici"];

            TextBox1.Text = "Ok";
            Response.Redirect("Default.aspx");
        }
        else
        { TextBox1.Text = "Hatalı"; }
        scon.Close();
        scon.Dispose();
    }
}
```

EK3. Default.aspx:

```
<link href="konum.css" rel="stylesheet" type="text/css"
/>
    <title>KKÜ - E B Y S</title>
    <style type="text/css">
        #bosluk {padding:0px;margin:0px;
        }
        .bosluk {padding:0px;margin:0px;
        }
    </style>
    <script type="text/javascript">

        function getmainpopup(adres)
        {
            mainpopup.SetContentUrl(adres);
            mainpopup.Show();
        }
    </script>
</head>
    <body style="overflow:hidden">
        <form id="form1" runat="server">

    <dx:ASPxPopupControl ID="ASPxPopupControl1"
runat="server" ClientInstanceName="mainpopup"
EnableClientSideAPI="True" Height="126px"
RenderMode="Lightweight"  Maximized="True" Modal="True"
PopupAnimationType="None" ShowFooter="True"
CssClass="bosluk">
        <ContentStyle>
            <Paddings Padding="0px" />
        </ContentStyle>
        <ContentCollection>
<dx:PopupControlContentControl runat="server"
SupportsDisabledAttribute="True"></dx:PopupControlContent
Control>
</ContentCollection>
        </dx:ASPxPopupControl>
    <div style="border-width: 0px; padding: 0px; margin:
0px; height: 750px; width: 1300px">
        <iframe src="ustBar.aspx" name="ustFrame"
scrolling="no"
style="width:1344px;height:80px;resize:none;border:0px;pa
dding:0px;margin:0px; position: absolute; top:0px; left:
0px">
        </iframe>
```

```
        <iframe src="solMenu.aspx" name="solFrame"
scrolling="no"
style="width:237px;height:650px;resize:none;border:0px;pa
dding:0px;margin:0px; position: absolute; top:75px; left:
0px">
        </iframe>

        <iframe src="ilkSayfa.aspx" name="icerik"
style="overflow:scroll;
width:1140px;height:550px;resize:none;border:0px;padding:
0px;margin:0px; position: absolute; top:75px; left:
210px">
        </iframe>

    </div>
  </form>
</body>
</html>
```

EK4. Evrak Arama:

```
public partial class Arama_EvrakArama_Default :
System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            ASPxDSILKTARIH.Date = DateTime.Now;
            ASPxDSSONTARIH.Date =
DateTime.Now.AddDays(30);
            ASPXGRLUPKURUMLISTESI.Value = null;
            ASPXGRLUPKURUMLISTESI.NullText = "Seçiniz.";
            ASPXGRLUPBIRIMLISTESI.Value = null;
            ASPXGRLUPBIRIMLISTESI.NullText = "Seçiniz.";
            TXTSAYI.Text = "";
            TXTKONU.Text = "";
        }
    }
    protected void
ASPxCallbackPnlEvrakListe_Callback(object sender,
DevExpress.Web.ASPxClasses.CallbackEventArgsBase e)
    {
        LBLBILGI.Text = "";
        Stopwatch Zaman = new Stopwatch();
        Zaman.Start();
```

```
        string[] ParametreListesi=
e.Parameter.Split('é');
        int Tip=-1;
        string Sayi="";
        string Konu="";
        int Kurum=-1;
        DateTime IlkTarih=DateTime.Now;
        DateTime SonTarih=DateTime.Now;
        int Birim=-1;
        if (ParametreListesi.Count()>0)
        {
            Tip = Convert.ToInt32(ParametreListesi[0]);
        }
        if (ParametreListesi.Count()>1)
        {
              Sayi=ParametreListesi[1];
        }
        if (ParametreListesi.Count()>2)
        {
              Konu=ParametreListesi[2];
        }
       if (ParametreListesi.Count()>3)
        {
            if (ParametreListesi[3] != "null")
            {
                Kurum =
Convert.ToInt32(ParametreListesi[3]);
            }
        }
       if (ParametreListesi.Count() > 4)
       {
           if (ParametreListesi[4] != "null")
           {
               Birim =
Convert.ToInt32(ParametreListesi[4]);
           }
       }
       if (ParametreListesi.Count() > 5)
       {
           IlkTarih =
Convert.ToDateTime(ParametreListesi[5]);
       }
       if (ParametreListesi.Count() > 6)
       {
           SonTarih =
Convert.ToDateTime(ParametreListesi[6]);
       }
       int KayitSayisi=0;
```

```
        using (AramaIstemci YeniRaporIstemci = new
AramaIstemci())
        {
            DataTable Veri =
YeniRaporIstemci.GenelEvrakAra(Tip, Sayi, Konu, Kurum,
Birim, IlkTarih, SonTarih);
            Cache[this.ClientID + "Data"] = Veri;
            KayitSayisi=Veri.Rows.Count;
            ASPxGRDEVRAKLISTESI.DataSource = Veri;
            Zaman.Stop();
            ASPxGRDEVRAKLISTESI.DataBind();
        }
        int minutes = (int)Zaman.Elapsed.TotalMinutes;
        double fsec = 60 * (Zaman.Elapsed.TotalMinutes -
minutes);
        int sec = (int)fsec;
        double ms = 1000 * (fsec - sec);
        if (KayitSayisi > 0)
        {
            LBLBILGI.Text = string.Format("{0} Kayıt
Bulundu. Arama İşlemi {1}:{2:D2}.{3} Sürdü", KayitSayisi,
minutes, sec, ms);
        }
        else
        {
            LBLBILGI.Text = "Uygun Kayıt Bulunamadı.!";
        }
    }
       protected void
ASPxGRDEVRAKLISTESI_DataBinding(object sender, EventArgs
e)
    {
        if (Cache[this.ClientID + "Data"]!=null)
        {
            ASPxGRDEVRAKLISTESI.DataSource =
Cache[this.ClientID + "Data"];
        }
    }
}
```

EK5. ebysverigonder.ashx:

```
<%@ WebHandler Language="C#" Class="ebysverigonder" %>
using System;
using System.Web;
public class ebysverigonder : IHttpHandler,
System.Web.SessionState.IRequiresSessionState  {
```

```
    public HttpContext cnt;
    public string adi;
    public string soyadi;
    public string tcno;
    public string adres;
    public string kodu;
    public int gonderenyaz()
    {
        System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection(sabitler.constring);
        string komut;
        if (cnt.Session["kurumici"].ToString()=="0")
            komut = @"insert into others
(adi,soyadi,tcno,adres,ownerbirim) values
(@adi,@soyadi,@tcno,@adres,@ownerbirim);SELECT @@IDENTITY
as id;";
        else
            komut = @"insert into sahislar
(adi,soyadi,tcno,ownerbirim) values
(@adi,@soyadi,@tcno,@ownerbirim);SELECT @@IDENTITY as
id;";
        System.Data.SqlClient.SqlCommand com = new
System.Data.SqlClient.SqlCommand(komut, conn);
        com.Parameters.AddWithValue("@adi",adi);
        com.Parameters.AddWithValue("@soyadi",soyadi);
        com.Parameters.AddWithValue("@tcno",tcno);
        if (cnt.Session["kurumici"].ToString() == "0")
com.Parameters.AddWithValue("@adres", adres);

com.Parameters.AddWithValue("@ownerbirim",cnt.Session["an
abirim"].ToString());
        try
        {
            conn.Open();
            object o = com.ExecuteScalar();
            conn.Close();
            conn.Dispose();
            return Convert.ToInt32(o.ToString());
        }
        catch
        {
            conn.Close();
            conn.Dispose(); return 0;
        }
    }
    public int kurumyaz()
    {
```

```
        System.Data.SqlClient.SqlConnection conn = new
System.Data.SqlClient.SqlConnection(sabitler.constring);
        string komut;
        komut = @"insert into kurumlar
(kurumadi,idaribirimkimlikkodu) values
(@adi,@kodu);SELECT @@IDENTITY as id;";
        System.Data.SqlClient.SqlCommand com = new
System.Data.SqlClient.SqlCommand(komut, conn);
        com.Parameters.AddWithValue("@adi", adi);
        com.Parameters.AddWithValue("@kodu", kodu);
       try
        {
            conn.Open();
            object o = com.ExecuteScalar();
            conn.Close();
            conn.Dispose();
            return Convert.ToInt32(o.ToString());
        }
       catch
       {
           conn.Close();
           conn.Dispose();
           return 0;
       }
    }
    public void ProcessRequest (HttpContext context) {
     //   context.Response.ContentType = "text/plain";
     //   context.Response.Write("Hello World");
        cnt = context;
        int id =Convert.ToInt32(
context.Request.QueryString["id"].ToString());        if
(id == 1)
        {
            adi = context.Request.QueryString["adi"];
            soyadi =
context.Request.QueryString["soyad"];
            tcno = context.Request.QueryString["tcno"];
            adres = context.Request.QueryString["adres"];
            int a = 0;
            cnt.Session["kurumici"] = "0";
            a = gonderenyaz();
            context.Response.ContentType = "text/plain";
            context.Response.Write(a.ToString());
        }
        if (id == 2)
        {
            adi = context.Request.QueryString["adi"];
            kodu = context.Request.QueryString["kodu"];
```

```
            int a = 0;
            a = kurumyaz();
            context.Response.ContentType = "text/plain";
            context.Response.Write(a.ToString());
        }
    }
    public bool IsReusable {
        get {
            return false;
        }
    }
}
```

EK6. Gelenevrakyonet.aspx.cs

```
public partial class gelenevrakyonet : System.Web.UI.Page
{
    public static string tippver(object content)
    {
        if (content is DBNull) return "";
        return content.ToString();
            }
    public static Datalar.gelenevrakekran dt;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            int a =
Convert.ToInt32(Request.QueryString["id"].ToString());
            int b =
Convert.ToInt32(Request.QueryString["data"].ToString());
            Session["gelenevrakid"] = b.ToString();
            Session["gelenevrakid_kayitlar"] =
a.ToString();
            ASPxGridView2.DataBind();
            Datalar d = new Datalar();
            dt= d.gelenevrakdataver(a);
            ASPxFormLayout1_E1.Text =
dt.havaletipi.ToString();
            ASPxTextBox11.Text = dt.tarih;
            ASPxTextBox13.Text = dt.ekler;
            ASPxTextBox14.Text = dt.evragigonderen;
            ASPxTextBox16.Text = dt.evrak_kayitt;
            ASPxTextBox17.Text = dt.kaydeden;
            ASPxTextBox18.Text = dt.gonderen;
            ASPxTextBox19.Text = dt.standartdosyaplani;
            ASPxTextBox2.Text = dt.sayi;
```

```
            ASPxTextBox21.Text = dt.kurum;
            ASPxTextBox22.Text = dt.durum;
            ASPxTextBox23.Text = dt.gizlilik;
            ASPxTextBox24.Text = dt.tamamlamat;
            ASPxTextBox4.Text = dt.konusu;
            ASPxTextBox8.Text = dt.evraks.ToString();
            ASPxFormLayout1_E3.Text = dt.havaletarihi;
            ASPxFormLayout1_E4.Text = dt.havaleeden;
            ASPxFormLayout1_E5.Text = dt.havaleaciklama;
        //    ASPxButton1.Enabled =
dt.havaletipi.ToString() == "Cevap Bekliyor";
            Session["cevapid"] = (Int32)dt.cevapid;
            ASPxTextBox8.Text =
dt.evraks;//evrakkarsiliksayi
        }
    }
    protected void ASPxGridView1_RowInserting(object
sender, DevExpress.Web.Data.ASPxDataInsertingEventArgs e)
   {
        e.NewValues["ownerevrak"] =
Session["gelenevrakid_kayitlar"].ToString();

e.NewValues["ownerbirim"]=Session["anabirim"].ToString();
        e.NewValues["kaynak"] =
Session["kullaniciid"].ToString();
        e.NewValues["paraf"] = 0;//
        e.NewValues["parafim"] = "Yeni";//
        e.NewValues["tarih"] =
System.DateTime.Now.ToString();
        Control cc;
        cc =
ASPxGridView1.FindEditRowCellTemplateControl(ASPxGridView
1.Columns["durum"] as GridViewDataColumn,
"ASPxComboBox3");
        e.NewValues["durum"] = (cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value.ToString()
;
        cc=
ASPxGridView1.FindEditRowCellTemplateControl(ASPxGridView
1.Columns["tipp"] as GridViewDataColumn,
"ASPxComboBox2");
        int tip=Convert.ToInt32( (cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value)+1;
        //
         cc=
ASPxGridView1.FindEditRowCellTemplateControl(ASPxGridView
1.Columns["havaleedilen"] as GridViewDataColumn,
"ASPxComboBox1");
```

```
        e.NewValues["hedef"] = (cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value;
        e.NewValues["hedefbirim"] = (cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).SelectedItem.Get
Value("id");
        //if  (((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).SelectedItem.Get
Value("veritabani").ToString() == "1") && ((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).SelectedItem.Get
Value("ozerk").ToString()=="0"))
        //{
        //    Datalar d = new Datalar();
        //    int cevapid =
Convert.ToInt32(Session["cevapid"].ToString());
        //    e.NewValues["triggered_gelenevrak"] =
d.yenigelenevrakver((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value.ToString()
, Session["gelenevrakid"].ToString(),
Session["anaevrak"].ToString(),tip,cevapid).ToString();
        //}
    }
    protected void
ASPxGridView1_CommandButtonInitialize(object sender,
DevExpress.Web.ASPxGridView.ASPxGridViewCommandButtonEven
tArgs e)
    {
        if (e.VisibleIndex == -1) return;
        if (e.IsEditingRow) return;
        switch (e.ButtonType)
        {
            case ColumnCommandButtonType.Edit:
                e.Visible =
EditButtonVisibleCriteria((ASPxGridView)sender,
e.VisibleIndex);
                break;
            case ColumnCommandButtonType.Delete:
                e.Visible =
DeleteButtonVisibleCriteria((ASPxGridView)sender,
e.VisibleIndex);
                break;
        }
    }
         public bool
EditButtonVisibleCriteria(ASPxGridView grid, int
visibleIndex) {
        object row = grid.GetRow(visibleIndex);
        bool b;
```

```
        b=
((DataRowView)row)["kaynak"].ToString().Contains(Session[
"kullaniciid"].ToString()) &&
(((DataRowView)row)["paraf"].ToString() == "0");
        b = b ||
(((DataRowView)row)["hedef"].ToString().Contains(Session[
"kullaniciid"].ToString()) &&
(((DataRowView)row)["paraf"].ToString() != "3"));
        return b;
    }
    public bool DeleteButtonVisibleCriteria(ASPxGridView
grid, int visibleIndex) {
        object row = grid.GetRow(visibleIndex);
        return
((DataRowView)row)["kaynak"].ToString().Contains(Session[
"kullaniciid"].ToString()) &&
(((DataRowView)row)["paraf"].ToString()=="0");
    }
    /*
    protected void
ASPxUploadControl1_FileUploadComplete(object sender,
DevExpress.Web.ASPxUploadControl.FileUploadCompleteEventA
rgs e)
    {
        string s;
        s = Server.MapPath("~/Ekler/") +
Session["birimid"].ToString() + "/"
+Session["gelenevrakid"].ToString()+"_"+
e.UploadedFile.FileName;
        ASPxUploadControl1.UploadedFiles[0].SaveAs(@s);
        SqlConnection conn = new
SqlConnection(sabitler.constring);
        string komut = @"insert into ekler
(path,ownerevrak,dosyaadi,ownerbirim,tarih,ownerpersonel)
values(@path,@ownerevrak,@dosyaadi,@ownerbirim,getdate(),
@ownerpersonel)";
        SqlCommand com = new SqlCommand(komut, conn);
        com.Parameters.AddWithValue("@path", "/Ekler/" +
Session["birimid"].ToString());

com.Parameters.AddWithValue("@ownerevrak",Session["gelene
vrakid"].ToString());
        com.Parameters.AddWithValue("@dosyaadi",
Session["gelenevrakid"].ToString()+"_"+e.UploadedFile.Fil
eName);

com.Parameters.AddWithValue("@ownerbirim",Session["birimi
d"].ToString());
```

```
        com.Parameters.AddWithValue("@ownerpersonel",
Session["kullaniciid"].ToString());
        conn.Open();
        com.ExecuteNonQuery();
        conn.Close();
        conn.Dispose();
    }*/
    protected void ASPxGridView2_CustomCallback(object
sender, ASPxGridViewCustomCallbackEventArgs e)
    {
        ASPxGridView2.DataBind();
    }
    protected void
ASPxGridView2_CustomDataCallback(object sender,
ASPxGridViewCustomDataCallbackEventArgs e)
    {
        ASPxGridView2.DataBind();
    }
    protected void
ASPxComboBox1_ItemsRequestedByFilterCondition(object
source,
DevExpress.Web.ASPxEditors.ListEditItemsRequestedByFilter
ConditionEventArgs e)
    {
    }
    protected void ASPxGridView1_RowUpdating(object
sender, DevExpress.Web.Data.ASPxDataUpdatingEventArgs e)
    {
        if
((e.NewValues["kaynak"].ToString()!=Session["kullaniciid"
].ToString())  ||  (e.OldValues["durum"].ToString() !=
"0") )
            e.NewValues["aciklama"] =
e.OldValues["aciklama"];
        e.NewValues["hedef"] = e.OldValues["hedef"];
        e.NewValues["ownerevrak"] =
e.OldValues["ownerevrak"];
        e.NewValues["ownerbirim"] =
e.OldValues["ownerbirim"];
        e.NewValues["kaynak"] = e.OldValues["kaynak"];
        e.NewValues["hedef"] = e.OldValues["hedef"];
        e.NewValues["triggered_gelenevrak"] =
e.OldValues["triggered_gelenevrak"];
        e.NewValues["durum"] = e.OldValues["durum"];
        if (e.NewValues["hedef"].ToString() ==
Session["kullaniciid"].ToString())
        {
```

```
            Control cc =
ASPxGridView1.FindEditRowCellTemplateControl(ASPxGridView
1.Columns["durum"] as GridViewDataColumn,
"ASPxComboBox3");
            e.NewValues["durum"] = (cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value.ToString()
;
        }
        else { e.NewValues["durum"] =
e.OldValues["durum"]; }
    }
    protected void ASPxGridView1_RowDeleting(object
sender, DevExpress.Web.Data.ASPxDataDeletingEventArgs e)
    {
    }
    protected void ASPxGridView1_RowInserted(object
sender, DevExpress.Web.Data.ASPxDataInsertedEventArgs e)
    {
    }
    protected void
ASPxGridView2_CommandButtonInitialize(object sender,
ASPxGridViewCommandButtonEventArgs e)
    {
        if (e.VisibleIndex == -1) return;
        if (e.IsEditingRow) return;
        switch (e.ButtonType)
        {
            case ColumnCommandButtonType.Delete:
                e.Visible =
DeleteButtonVisibleCriteria2((ASPxGridView)sender,
e.VisibleIndex);
                break;
            case ColumnCommandButtonType.Edit:
                 e.Visible =
DeleteButtonVisibleCriteria2((ASPxGridView)sender,
e.VisibleIndex);
                break;
        }
    }

    public bool DeleteButtonVisibleCriteria2(ASPxGridView
grid, int visibleIndex)
    {
        object row = grid.GetRow(visibleIndex);
        try
        {
            return
(((DataRowView)row)["ownerpersonel"].ToString() ==
```

```
(Session["kullaniciid"].ToString()) &&
((Convert.ToDateTime(((DataRowView)row)["tarih"])) <
(System.DateTime.Now.AddMinutes(-25))));
        }
        catch { return false; }
    }
    protected void ObjectDataSource1_Updating(object
sender, ObjectDataSourceMethodEventArgs e)
    {
        e.InputParameters["id"]=Session["ekid"];
        e.InputParameters["aciklama"] =
Session["aciklama"];
    }
    protected void ASPxGridView2_RowUpdating(object
sender, DevExpress.Web.Data.ASPxDataUpdatingEventArgs e)
    {
        Session["ekid"]=e.NewValues["id"];
        Session["aciklama"]=e.NewValues["aciklama"];
    }
    protected void ASPxButton1_Click(object sender,
EventArgs e)
    {
        Session["imzalandi"] = (int)0;
        evraklar.standartdosyaplani = dt.sdandartdp;
        evraklar.cevapid = dt.cevapid;
        evraklar.havaletipi = dt.havaletipi_;
        evraklar.konusu = dt.konusu;
        evraklar.ekler = dt.ekler;
        evraklar.durum = dt.durum_;
        evraklar.gizlilik = dt.gizlilik_;
    //    Session["gelenevrakid"] =
Session["gidenevrakid_kayitlar"];
        Response.Redirect("Default4.aspx?gelenecevap=" +
Session["gelenevrakid_kayitlar"] + "&&cevapid=" +
Session["cevapid"]);
    }
    protected void ASPxGridView1_InitNewRow(object
sender, DevExpress.Web.Data.ASPxDataInitNewRowEventArgs
e)
    {
        e.NewValues["paraf"] = 0;//
        e.NewValues["parafim"] = "Yeni";//
    }
    protected void ASPxGridView1_CustomCallback(object
sender, ASPxGridViewCustomCallbackEventArgs e)
    {
            for (int a = 0; a <
ASPxGridView1.VisibleRowCount; a++)
```

```
        {
            bool database =
(bool)ASPxGridView1.GetRowValues(a,"veritabani");
            if (!database ) continue;
            int triggeredevrak = 0;
            try
            {
               triggeredevrak=
(Int32)ASPxGridView1.GetRowValues(a,
"triggered_gelenevrak"); }
             catch {triggeredevrak = 0;  }
            if (triggeredevrak>0) continue;
            int anabirim =
(Int32)ASPxGridView1.GetRowValues(a, "hedefbirim");
            if (anabirim == 1) continue;  // rektörlüğe
havale

evraklar.evraksayisi=evraklar.evraksayisiuret(anabirim,
1);  // gelen evrak
            evraklar.anaevrak =
Convert.ToInt32(Session["gelenevrakid"]);
            evraklar.ownerevrak =
Convert.ToInt32(Session["gelenevrakid_kayitlar"]);
            evraklar.anabirim = anabirim;
            evraklar.gelenmi = 1;
            evraklar.cevapid = (Int32)dt.cevapid;
            evraklar.kayitlaraekle();
            evraklar.havaletipi =
Convert.ToInt32(ASPxGridView1.GetRowValues(a,
"havaletipi"));

            int id =
Convert.ToInt32(ASPxGridView1.GetRowValues(a, "id"));


evraklar.havalelere_gelenevrakbilgisikaydet(id,evraklar.k
ayitlar
        }
        ASPxGridView1.DataBind();
        //if  (((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).SelectedItem.Get
Value("veritabani").ToString() == "1") && ((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).SelectedItem.Get
Value("ozerk").ToString()=="0"))
        //{
        //    Datalar d = new Datalar();
        //    int cevapid =
Convert.ToInt32(Session["cevapid"].ToString());
```

```
        //    e.NewValues["triggered_gelenevrak"] =
d.yenigelenevrakver((cc as
DevExpress.Web.ASPxEditors.ASPxComboBox).Value.ToString()
, Session["gelenevrakid"].ToString(),
Session["anaevrak"].ToString(),tip,cevapid).ToString();
        //}
    }
}
```

ÖZGEÇMİŞ

Adı Soyadı : Mahmut ÜNVER

Doğum Tarihi : 02.05.1970

Yabancı Dil : İngilizce (ÜDS:70)

Eğitim Durumu : (Kurum ve Yıl)

Lisans : Gazi Üniversitesi, Teknik Eğitim Fakültesi, Bilgisayar Sistemleri Eğitimi, 1993

Yüksek Lisans : Gazi Üniversitesi, Bilişim Enstitüsü, Bilgisayar Eğitimi, 2010

Çalıştığı Kurum/Kurumlar ve Yıl/Yıllar:

Gazi Üniversitesi, 1994-1998

Kırıkkale Üniversitesi: 1999-

Yayınları (SCI) :

1. Developing a File System Structure to Solve Healthy Big Data Storage and Archiving Problems Using a Distributed File System, Applied Science, 2018, 8(6), 913; https://doi.org/10.3390/app8060913

Yayınları (Diğer) :

Araştırma Alanları : Big Data, Dağıtık Dosya Sistemleri, Algoritma, Web Programlama,