Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-15
Sayı-2
2022

# Büyük Veri Tabanlı Arşivleme Yönetim Sistemi

## Big Data Based Archiving Management System

Ayşegül Şenol ÇALIM
*Türkiye Finans Katılım Bankası*
İstanbul, Türkiye
aysegul.senolcalim@turkiyefinans.com.tr
ORCID: 0000-0003-0062-3032

Gökay TUNA
*Türkiye Finans Katılım Bankası*
İstanbul, Türkiye
gokay.tuna@turkiyefinans.com.tr
ORCID: 0000-0003-4305-9451

## Öz

*Yeni ürünlerin hizmete sunulması, müşteri sayısının hızla artması, denetim iz kayıtları gibi zorunlu olarak tutulması gereken verilerin boyutlarının fazla olması nedeniyle banka gibi kurumlarda veri büyüklüğü hızlı bir şekilde artmaktadır. Bu veriler mevcut sistemlerde yıllarca kaldığı zaman sistemleri ve uygulamaları ağırlaştırmakta, yedekleme ve sistem bakımı gibi operasyonel işlemlerin maliyetlerini arttırmaktadır. Bütün bu sorunlar için verilerin sınıflandırılarak erişim sıklığına göre kategorize edilmesi, sınıflandırılan verilerin anlık erişim gereksinimi olmayanlarının ikincil ve daha az maliyetli sistemlere taşınarak arşivlenmesi ve arşivlendikten sonra kaynak sistemden silinmesi gerekmektedir. Bu gereksinimleri karşılamak için yazılan büyük veri tabanlı arşivleme yönetim sistemi, bir yazılım ürünü olarak geliştirilmiştir. Yapısal ve yapısal olmayan verilerin Hadoop ekosisteminde arşivlenmesi verilere daha etkin erişim sağlayacak ve daha ucuz saklama maliyetleri getirecektir. Bu bağlamda yapılan çalışmada veri erişim katmanı, hizmet ve uygulama katmanından oluşan verileri, HDFS (Hadoop Distributed File System) dosya sistemi yapısında dağıtık olarak üç kopya halinde tutan ve fiziksel sunucular üzerinde sanallaştırma teknolojileri kullanılarak kurulan bir büyük veri tabanlı arşivleme yönetim sistemi geliştirilmiştir. Sonrasında ise ilişkisel veri tabanlarındaki yapısal tabloların erişimi ve aktarımı için Sqoop, yapısal olmayan kaynaklardan yapılan aktarımların gerçekleştirilmesi ve zamanlanması için Nifi araçları kullanılmıştır.*

*İlişkisel veri tabanlarından büyük veri arşivleme sistemine aktarılan ve HDFS dosya yapısında tutulan verilerde hacimsel büyüklükte çok büyük oranda azalma gerçekleşmiştir. Veri erişim katmanı üzerinden yapılan veri sorgulama işlemlerinde ise özellikle sayısal olmayan verilerin erişiminde yüksek performans artışları gözlemlenmiştir.*

Gönderme, düzeltme ve kabul tarihi: 11.10.2021 - 07.08.2022 - 19.10.2022
Makale türü: Araştırma

*Bu çalışmada, büyük veri arşivleme ve veri analitiği kavramları incelenmiş ve bu kavramlar üzerine yapılan çalışmalar araştırılmıştır. Bu kapsamda gerçekleştirilen kaynak taramasından çıkarılan bilgiler neticesinde; veri büyüklüğünün ve veriye erişim hızı gereksinimin hızla arttığı kurumlarda kullanılabilecek olan, yüksek hız, yüksek verimlilik, daha az maliyet ve daha fazla çeşitlilik gereksinimlerini karşılayan bir arşivleme yönetim sistemi büyük veri platformu üzerinde gerçekleştirilmiştir.*

**Anahtar Sözcükler:** Hadoop, büyük veri, arşivleme, veri arşivleme yönetimi, yapısal olmayan veri arşivleme, yapısal veri arşivleme, büyük veri aşivleme

## Abstract

*The size of data in institutions such as banks is increasing rapidly due to the fact that the number of new products are put into service, the number of customers are increased, the number of new applications are put into use due to regulations, and the data that must be kept compulsory such as audit trail records are excessive. When these data remain in existing systems for years, systems and applications become heavy, and the costs of operational processes such as backup and system maintenance increase. For all these problems, the data should be classified and categorized according to the frequency of access, those that do not need instant access to the categorized data should be archived by moving them to secondary and less costly systems and deleted from the source system. The big data-based archiving management system will be developed as a software product, providing more effective access to structural or unstructured data to be archived in the Hadoop ecosystem and bringing cheaper storage costs.*

*In this context, a big data-based archiving management system, which consists of data access layer, service and application layer, keeps data in three copies distributed in HDFS (Hadoop Distributed File System) file system structure that established by using virtualization technologies on*

*physical servers is developed. Afterwards, Sqoop for accessing and transferring structural tables in relational databases and Nifi for performing and scheduling transfers from non-structural sources were used.*

*The volume of data transferred from relational databases to the big data archiving system which is kept in the HDFS file structure, has decreased significantly. High performance increases have been observed in data query operations made over the data access layer, especially in the access of non-numerical data.*

*In this study, the concepts of big data archiving and data analytics were examined and studies on these concepts were investigated. As a result of the information obtained from the literature review carried out in this context; An archiving management system has been implemented on a big data platform, which can be used in institutions where the size of data and the need for data access speed are increasing rapidly, meeting the needs of high speed, high efficiency, less cost and greater diversity.*

**Keywords**: Hadoop, big data, archiving, data archiving management, unstructured data archiving, structred data archiving, big data archiving

## 1. Giriş

Arşivleme herhangi bir formattaki, etkin olmayan ya da erişim sıklığı az olup düzenli olarak kullanılmayan verilerin güvenli, tutarlı, hızlı ve erişebilir olarak, gerçek zamanlı uygulamaların veri tabanlarından farklı bir depolama alanında saklanması şeklinde tanımlanabilir.

Buna karşın arşiv verileri resmi ve resmi olmayan kurumlar için, yasal talepler, uzun vadeli analiz, raporlama ve kanıt ihtiyaçları için kullanılmak üzere önemini korur.

Bu tarz verilerin canlı ortamlardan alınıp daha ucuz ortamlarda (yedek teyp vb.) saklanması olanaklı olsa da bu durum iş gereksinimlerinin zamanında karşılanmasına olanak vermemektedir.

Bu nedenle seyrek kullanılmasına karşın erişme gereksinimi olan bu tip veriler, genellikle sistemlerin canlı ortamlarında tutulmaktadır. Bu durum da hem veri tabanı başarımı hem de depolama anlamında yüksek maliyetler ortaya çıkarmaktadır. Bu şekildeki maliyetlerin düşürülmesi ve arşiv verilerinin gerçek zamanlı uygulamalar üzerindeki yükünün azaltılması gerekliliği, arşivleme yönetim sistemlerine olan gereksinimin ortaya çıkmasına yol açmıştır. Bu kapsamda, arşivleme yönetim sistemlerinin çözmesi gereken en büyük sorunlar:

- daha az kullanılan büyük miktardaki verinin daha az maliyetli olarak saklanması,
- yeni teknolojiler kullanılarak veri erişim ve yedekleme başarımının yükseltilmesi,
- arşivlenecek olan nesne ve tabloların modüler olarak da arşivlenebilmesidir.

Günümüz arşivleme yaklaşımları çoğunlukla tüm veri tabanı tablolarının ve üst verilerinin birleşik bir şekilde başka bir yapısal ve ilişkisel veri tabanı platformuna taşınması şeklindedir. Bu yaklaşımlar ile geliştirilmiş olan RODADB [1], SIARD [2] ve CHRONOS [3] araçları incelendiğinde ortak olarak uygulanan yöntemin, ilişkisel veri tabanındaki tüm verinin XML [4] veya CSV [5] gibi sıkıştırılmış bir formata dönüştürülerek saklanması ve ihtiyaç halinde yeniden ilişkisel veri tabanı formatına çevrilmesi şeklinde olduğu söylenebilir.

Örnek olarak yapılan çalışmalarda İsviçre Federal Arşivleri [6] tarafından geliştirilen SIARD incelendiğinde ilişkisel veri tabanını XML ve XSD [7] format çiftlerine dönüştürdükten sonra ZIP64 [8] standardında sıkıştırılmış bir SIARD dosya formatında [9] arşivlediği belirlenmiştir [10]. Dönüştürülen dosya formatına erişimin SIARD Suite [11] adlı bir yardımcı araç kullanılarak ya da yeniden ilişkisel bir veri tabanına çevrilerek sağlanabildiği; ancak SIARD aracının veri tabanındaki yordamlar, birincil anahtarlar gibi nesneleri arşivlemediği ve veri bütünlüğünde kayıplara neden olabildiği kullanıcı deneyimlerine dayanılarak belirtilmiştir [12]. Ayrıca 10 TB üzeri veri tabanlarının arşivlenmesi esnasında başarım ve yeniden kullanım sıkıntıları yaşanabildiği daha önce yapılan durum çalışmalarında deneyimlenmiştir [13].

Günümüzde geliştirilmiş ve kullanılmakta olan arşivleme yönetim sistemlerinin veri sıkıştırma, veri taşıma, arşivleme, arşivlenen veriye erişme hususlarında yadsınamayacak bir fayda sağlamasının yanı sıra performans, maliyet ve tutarlılık konularında gözlemlenebilir eksikliklerinin olduğu bunun yanında nesnelerin ve tabloların bağımsız olarak arşivlenmesine olanak vermediği gözlemlenmiştir.

Bu nedenle arşivleme sistemleri veri yaşam döngüsünün bir parçası olarak ele alınmalı; çoğunluğu yapısal olarak değerlendirilebilecek olan kurum verilerinin, klasik ilişkisel veri tabanları sistemlerinde tutulup kurumlara ek lisans maliyetlerine neden olmasının önüne geçilmelidir. Ek olarak sınıflandırma sonucunda arşivlenecek olan nesnelerin ve tabloların tekil olarak arşivlenebilmesi, bu işlemlerin düzenli olarak zamanlanmış ve tekrarlayan bir şekilde yapılması gerekmektedir.

Belirtilen gereksinimlerin karşılanması için geliştirilen "büyük veri tabanlı arşivleme yönetim sistemi" yapısal ve yapısal olmayan verilerin;

- yönetimsel veri sınıflandırmasının yapılmasını,
- veri yaşam döngüsünün yönetimini,
- kayıt altına alınmasını ve yasal düzenleyici kurallara uyumunu,
- yönetimsel olarak gereksinim duyulmayan verilerin arşivlenmesini sağlayan bir veri arşivleme platformudur.

Büyük veri tabanlı arşivleme yönetim sistemi ile verilerin sınıflandırılarak erişim sıklığına göre sınıflandırılması, sınıflandırılan verilerden anlık erişim gereksinimi olmayanlarının ikincil ve daha az maliyetli sistemlere taşınması, aynı verinin daha küçük boyutlarda arşivlenmesi, veri erişim başarımının artırılması ve kaynak sistemlerden silinmesi olarak özetlenecek bir yapının kurulması hedeflenmiştir. Bu sistem bir yazılım ürünü olarak geliştirilmiştir.

## 2. Büyük Veri Tabanlı Arşivleme Yönetim Sistemi Uygulaması

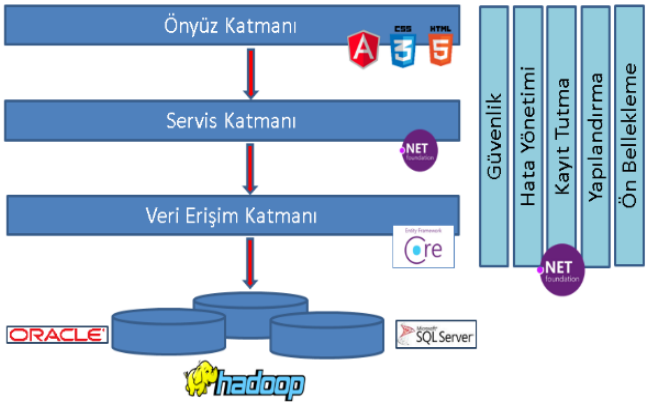
Bu bölümde arşivleme yönetim sistemi uygulamasının teknolojik bileşenleri hakkında ayrıntılı bilgi verilerek uygulamanın mimarisi anlatılacaktır.

Uygulama veri arşivleme, veri aktarımı ve veri erişimine olanak sağlayacak olduğundan katmanlı mimari ilkeleri göz önüne alınarak geliştirilmiştir. Geliştirilen uygulamada büyük veri platformu uygulamanın ana omurgasını oluşturmaktadır. Önyüz katmanı, servis katmanı ve veri erişim katmanından oluşan uygulama mimarisi detaylı şekilde açıklanmıştır.

### 2.1 Uygulama Mimarisi

Uygulama mimarisi Şekil-1'de gösterilmiştir. Mimari açıdan uygulama 3 katmandan oluşmaktadır.

- Önyüz Katmanı
- Servis Katmanı
- Veri Erişim Katmanı




**Şekil-1:** Uygulama Mimarisi

Önyüz katmanında web tabanlı ekranlar geliştirilmiştir. Bu katmanda Angular, HTML5 ve CSS3 teknolojileri kullanılmıştır. Angular [14] Google tarafından geliştirilen, yazılımcıların hayatını kolaylaştıran javascript kütüphaneleri içeren bir ön yüz çerçevesidir. Geliştirilen bu ekranlar ile son kullanıcının veri okuma, görev tanımlama ve oluşturulan görevlerin takip edilmesi gibi işlemleri yapabilmesinin yanı sıra uygulama yöneticilerinin de kullanıcı rol, grup ve yetki tanımlamaları yapabilmeleri amaçlanmıştır.

Hizmet katmanında farklı işletim sistemleri üzerinde çalışabilen, açık kaynak kodlu, Microsoft tarafından desteklenen, farklı mimarilere uyumlu olarak uygulanabilen bir platform olan .netCore [15] teknolojisi kullanılmıştır. Bu katman ile önyüz katmanından yapılan isteklerin karşılanması ve veri erişim katmanı ile bütünleşmesi amaçlanmıştır.

Aynı amaç doğrultusunda veri erişim katmanında ise .netCore Entity Framework teknolojisi kullanılarak veri tabanı teknolojisinden bağımsız olarak servis katmanı ile veri alışverişinin gerçekleştirilmesi hedeflenmiştir.

Uygulamanın her katmanında genel altyapı çerçeveleri sunulmaktadır. Bunlar güvenlik, hata yönetimi, kayıt tutma, yapılandırma ve ön bellekleme fonksiyonlarıdır. Bu genel alt yapı fonksiyonları için de yetkilendirme ve uygulama güvenliği anlamında gelişmiş çözümler sunan .netCore teknolojisi kullanılmıştır.

Veri tabanı olarak gereksinimler doğrultusunda Oracle ve MsSQL veri tabanlarından veri okumanın yapılarak Hadoop [16] büyük veri ortamına aktarılması sağlanmıştır; ancak uygulama gerektiği durumlarda diğer veri tabanlarına da kolaylıkla erişim sağlayabilecek şekilde tasarlanmıştır.

### 2.2 Alt Yapı Mimarisi

Büyük veri tabanlı arşivleme yönetim sistemi projesi Şekil-2'de gösterilen sunucu ve kaynaklar ile tasarlanmış ve kurulmuştur. Bu ortamın sınama ortamı sanal sunucular üzerinde konumlandırılmış, üretim ortamı için ise içerisinde çok sayıda, büyük hacimli yerel diskin olduğu fiziksel sunucular kullanılmıştır. Bu fiziksel sunucular üzerine sanallaştırma teknolojileri kurularak, her fiziksel sunucu üzerinde atanmış bir sanal sunucu çalışacak şekilde kurulumlar yapılmıştır. Bu sanallaştırma yöntemi ile yedekleme, bakım ve operasyonel işlemlerin kolaylaştırılması hedeflenmiştir.

Başarım tarafında, paylaşımlı sanallaştırma katmanları kullanmak yerine amaca uygun atanmış sanal sunucular ve kaynaklar ile sistem kurulumları yapılmış; böylelikle gelecekte istenebilecek büyüme ve başarım gereksinimleri için ölçeklenebilir bir yapı kurgulanmıştır. Büyük veri arşivleme altyapısında kullanılan fiziksel sunuculara bağlı diskler ve HDFS [17] teknolojisiyle paralel veri yazma ve okuma işlemi sayesinde veriye erişimde performans elde edilmesi hedeflenmiştir.

Maliyet tarafında, büyük hacme sahip verilerin bulunduğu pahalı disk havuzlarında bulunan verilerin büyük veri arşivleme platformuna taşınması ile ciddi kazanımlar elde edilmiştir. Yapılan çalışmalar öncesinde geniş hacimli veriler kurumsal altyapıda kullanılan ve birçok iş kritik sisteme hizmet eden pahalı SSD (Solid State Drive – Yarı iletken disk) disk havuzları üzerinde tutulmakta iken, büyük veri altyapısının devreye alınması ile maliyet açısından kazanımların sürekliliği sağlanmıştır.

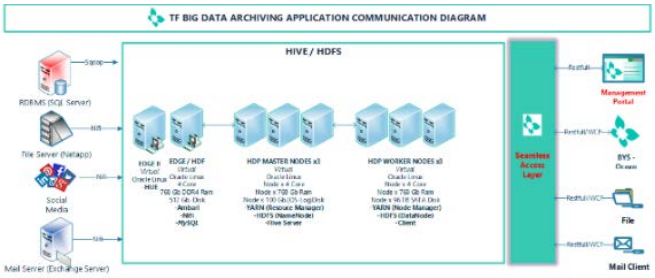
Büyük veri arşivleme platformu altyapısında büyük hacimli, maliyet olarak düşük, kendi içerisinde yedekleme ihtiyacı bulunmayan sunucu üzerinde barınan yerel disk sistemleri kullanılmıştır.

Yerel yedeklilik, birbirine paralel çalışan ve yedekleme sistemini kendi içerisinde barındıran HDFS ile sağlanmıştır.


**Şekil-2:** Alt Yapı Mimarisi

## 2.3 Tümleşik Mimari

Mimarinin bileşenlerinin birbirleri ile bütünleşmesinde kimlik doğrulama için kurumsal yapıda kullanılan etkin dizin (LDAP) bileşenleri kullanılmış; veri tabanları, sosyal medya verileri dosya sistemleri ve e-posta sunucularında bulunan veriler Şekil-3'te gösterilen bütünleşme yöntemleri ile bütünleştirilmiştir. LDAP kullanımı ayrı bir protokol geliştirme ihtiyacını ortadan kaldırmış, mevcut kurumsal yetkilendirme protokolü dahilinde, ilişkisel veri tabanı ve kurumsal sistemler üzerinde yetkilendirme yapılmış olan kullanıcı ve kullanıcı gruplarının, arşivlenen verilerinin büyük veri tabanlı arşivleme yönetim sistemi özelinde de kolaylıkla yetkilendirilebilmesini sağlamıştır.




**Şekil-3:** Tümleşik Mimari

Banka ağı içerisinde bulunan ana bankacılık sistemleri ile bütünleşme için REST mimarisi yanında WCF (Windows Communication Foundation) [18] mimarisi de kullanılmıştır.

Veri kaynaklarından veri akışı işlemleri ise Apache Nifi [19] ve Apache Sqoop [20] gibi büyük veri mimarisi bileşenleri kullanılarak sağlanmıştır. Apache ekosisteminin birer parçası olarak, sistemler arasındaki veri bütünleşmesinde kullanılabilen bu araçların ikisinin birden kullanılma nedeni farklı araçların daha etkin olan özelliklerinin kullanılarak daha verimli bir veri aktarım mekanizmasının kurulmasıdır. Örneğin; İlişkisel veri tabanlarından HDFS'e yapılan veri aktarımlarında paralellik sağlama açısından Sqoop tercih edilirken, yapısal olmayan daha karmaşık veri akışlarının sağlanmasında Nifi aracı kullanılmıştır. Ayrıca Nifi, veri aktarımlarının zamanlanması ve tekrar edilebilir çalışmasını sağlayacak olan kolay fonksiyonlar sunmaktadır.

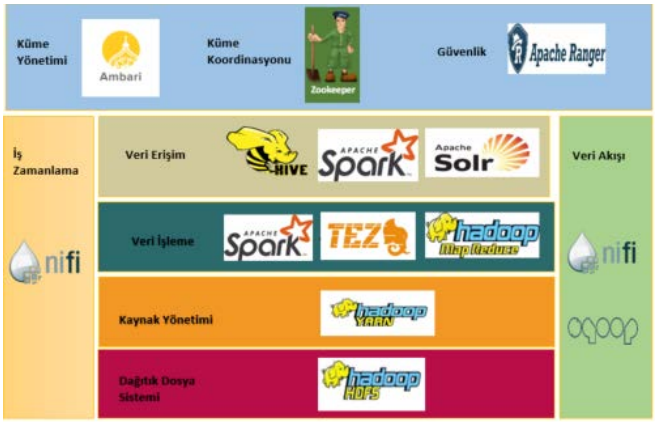
# 3. Uygulama Büyük Veri Ekosistemi

Bu bölümde büyük veri arşivleme yönetim sistemi uygulamasının kullandığı büyük veri bileşenlerinden ve bu bileşenlerin birlikte nasıl çalıştığından bahsedilmiştir.

Uygulamada kullanılan büyük veri ekosistemini oluşturan bileşenler Şekil-4'te gösterilmiştir.

## 3.1 Küme Yönetimi, Koordinasyonu ve Güvenliği

Hadoop kümeleri düğüm adı verilen, birbirine ağ üzerinde bağlı birden fazla bilgisayarın tek bir sistem olarak çalıştığı; yapılandırılmış, yapılandırılmamış ya da yarı yapılandırılmış büyük miktardaki verilerin işlenmesine olanak sağlayan bileşenlerdir. Hadoop kümelerinde büyük hacimli işler, küme üzerindeki ana düğüm (Master Node) [21] adı verilen yönetici düğümler ve ana düğümler tarafından yönetilen çalışan düğümler (Worker Node) [22] tarafından paralel olarak gerçekleştirilir [23].




**Şekil-4:** Uygulama Büyük Veri Ekosistemi

Hadoop küme yapısının paralel işleme, büyük miktardaki veriyi dağıtık olarak saklama - verinin farklı düğümlerde birden fazla kopyasının olması - ve hata toleransını azaltması şeklinde belirtilebilecek avantajları bulunmaktadır. Bu dağıtık yapının izlenmesi, giderek büyüyen veri hacmi ve işleme gereksinimin ölçeklenebilir bir şekilde yönetilmesi gereklidir. Bu amaçlar için Ambari [24] bu kümenin yönetiminde kullanılmıştır.

Ambari, Hortonworks Data Platformu'nun [25] bir parçası olarak çalışan, Hadoop kümesinde bulunan düğümlerin ve hizmetlerin yönetilmesini, izlenmesini ve ayarlarının yapılmasını sağlayan açık kaynaklı bir küme yönetimi platformdur. Kolay kullanımlı web arayüzünün olması, MapReduce [26], HDFS, Hive [27] gibi birçok Hadoop bileşenini desteklemesi ve aynı zamanda platform bünyesindeki Apache Ranger [28] bileşeni sayesinde güvenlik desteğini sağlaması nedeni ile çalışmamızda küme yönetim aracı olarak tercih edilmiştir.

Ambari tarafından yönetilen kümede bulunan hizmetlerin birbirleri ile koordinasyonunu ve senkronize bir şekilde çalışmasını da sağlamak gerekmektedir. Bu amaçla Yahoo tarafından geliştirilmiş olan ZooKeeper [29] kullanılmıştır. Zookeper, büyük veri platformunda bulunan HDFS, NameNode gibi kritik servislerin yüksek kullanılabilir durumda olmasını sağlar.

Kümenin güvenliği için ise yukarıda değinilen Apache Ranger kullanılmıştır. Ranger, Hadoop platformunda veri güvenliği, izleme ve yönetme alanlarında işlevler gerçekleştirir. Büyük veri kümesinde bulunan verilere erişim izinlerinin düzenlenmesini ve erişimlerin kayıt altına alınmasını da Ranger gerçekleştirir.

Özetle Şekil-4'de de görüldüğü üzere çalışmamızda kümenin yönetiminde Apache Ambari, kümedeki hizmetlerin koordinasyonunda Zookeper ve kümenin güvenliğinde Apache Ranger kullanılmıştır.

## 3.2 Dağıtık Dosya Sistemi

Büyük veri sisteminin varsayılan dosya sistemi olan HDFS; ilişkisel veri tabanlarından arşivleme amaçlı olarak aktarılan büyük veri platformu verilerinin yanı sıra sosyal medya verisi gibi yapısal olmayan verilerin de dağıtık olarak tutulduğu bir depolama aracıdır.

HDFS'in karakteristik özelliği dosyayı yazarken birden fazla kopya şeklinde yazmasıdır. Hdfs'e yazılan bir dosyanın varsayılan kopya sayısı üçtür. Büyük veri sistemleri bir kere

yazılıp çok kez okunma amaçlı tasarlanmış sistemler olduğundan dolayı, HDFS'de bulunan veriler silinebilir; fakat güncellenemezler.

HDFS dağıtık çalışması ve bir veriyi kümede bulunan birden fazla sunucuya yazması gibi özellikleri ile veriye hızlı erişilmesine ve hata toleransının azaltılmasına olanak sağlar. Bu hızlı erişimde veri blok boyutlarının da önemi vardır. Varsayılan veri blok boyutu klasik veri tabanı sistemlerinde 8 Kilo byte (KB) iken; HDFS yapısında varsayılan veri blok boyutu 128 Megabayttır (MB). Bu sayede HDFS büyük boyutlu verilere hızlı erişim yapılmasına olanak sağlamaktadır. 128 MB varsayılan blok büyüklüğü gereksinimlere göre düzenlenebilmektedir.

Maliyet tarafında ise HDFS verileri saklamak için maliyeti yüksek ve kendinden yedekli sunuculara ihtiyaç duymamaktadır. Bu yönü ile de büyük boyutlu veriler için ideal bir veri saklama ortamıdır.

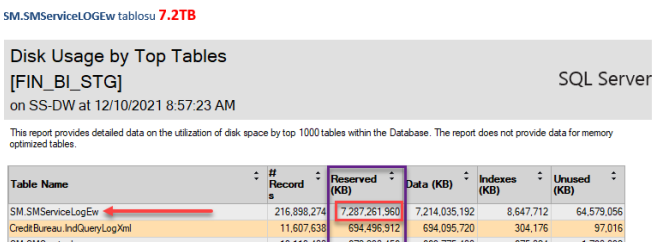
Örneğin; 10/12/2022 tarihinde MsSQL Server ilişkisel veri tabanı üzerinde bulunan 20 aylık servis log kayıtlarını içeren tablonun toplam disk kullanım hacmi Şekil 5'deki gibi 7.2 Terabayt (TB) olarak ölçülmüştür. İlgili tablonun yol açtığı depolama maliyeti, veri tabanı sunucusunda neden olduğu başarım maliyeti ve log tablosunun içerdiği XML verilerinde hızlı ve etkin bir şekilde yapılması gereken ayrıştırma işleminin zorunluluğu nedeniyle büyük veri tabanlı arşivleme yönetim sisteminde arşivlenme gerekliliği belirlenmiştir.

İlgili tablodaki tüm veriler, önyüz katmanında belirtilen "görev tanımlama" fonksiyonu kullanılarak ilerleyen bölümlerde daha ayrıntılı şekilde anlatılacak olan yöntemler ile blok boyutu 256 Mega byte (MB) olarak tanımlanmış olan HDFS dosya sistemine taşınmıştır.

HDFS üzerindeki veri, Şekil-6'daki gibi her ay için 1 adet olmak üzere toplam 20 adet alt dosyadan oluşan; her aylık alt dosyanın ortalama 400 MB'lık 120 adet yığından oluştuğu, aylık olarak bölünmüş bir yapıda arşivlenmiştir.

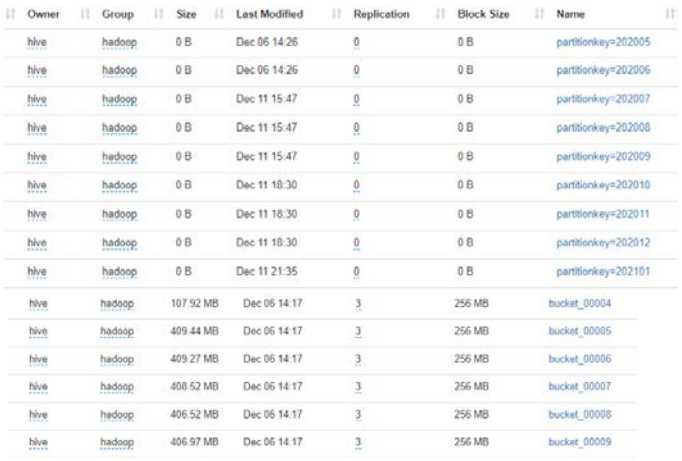
Yapılan ölçüm sonucunda bir kopyada tutulan veri boyutu 0,92 TB olarak belirlenmiş ve ilişkisel veri tabanında 7,2 TB disk kullanımı olan log tablosunun arşivlenmesinden %87 oranında bir depolama alanı kazanımı sağlanmıştır. Veri erişim sıklığı az olarak sınıflandırılan bu tablonun verileri, ilişkisel veri tabanından silinerek hem daha az maliyetli hem de daha az disk gereksinimi olan bir arşivleme sistemine taşınmıştır.

İlişkisel veri tabanları üzerinde geleneksel arşivleme yaklaşımlarından olan SIARD ve CHRONOS araçlarının kullanımı ile yapılan bir çalışmada elde edilen verilere göre; SIARD aracı kullanımı sonucunda elde edilen arşivin fiziksel büyüklüğü %338 oranında artarken, CHRONOS aracı kullanarak elde edilen arşivin fiziksel büyüklüğünde %41 oranında bir azalma ölçülmüştür [30].




**Şekil-5:** İlişkisel veri tabanı disk kullanımı

Daha önce yapılan bu çalışmalarda elde edilen veriler ile karşılaştırıldığında HDFS dosya sistemindeki verinin sağladığı fiziksel alan kazanımının her iki aracın sağladığı kazanımdan çok daha fazla olduğu sonucu çıkarılabilir. Aynı zamanda oluşturulan veri erişim, veri aktarım ve veri işleme bileşenlerinin sağladığı fonksiyonlar sayesinde - ilgili geleneksel arşivleme yöntemlerinin düşük başarım gösterdiği - nesnelerin zamanlı ve tekrarlayan şekilde arşivlenmesi, arşivlenen verilere güvenli, hızlı ve sürekli erişimin sağlanması hususlarında da oluşturulan uygulamanın daha verimli sonuçlar verdiği görülebilir.




**Şekil-6:** HDFS disk kullanımı

## 3.3 Dağıtık Kaynak Yönetimi

HDFS dosya sistemi, üzerinde yapılacak olan işlemlerin kaynak yönetiminin yapılabilmesi için bir kaynak yönetim sistemine gereksinim duyar. Bu noktada YARN [31] (Yet Another Resource Negotiator) büyük veri kümesinin kaynaklarının dağıtımını yapar. Büyük veri kümesinde yer alan servislerin hangi kaynakları ne kadar kullanacağının kararını verir.

Kaynakların nasıl zamanlanacağına karar veren, küme yöneticisinin kaynak kullanımını takip eden ve aynı zamanda kaynakların izin ve güvenlik ayarlarının da yönetimini yapan mekanizmaya kaynak yönetimi denir. YARN büyük veri kümesindeki ana düğüm ve çalışan düğümlerdeki kaynakların kullanımını zamanlar, izler ve gelen iş isteklerini kaynaklara dağıtır. Bir başka tanımla veri işleme sürecindeki tüm uygulamaların hangi kaynağı, ne zaman ve ne kadar meşgul edeceğinin yönetimini yapar.

## 3.4 Veri İşleme

Büyük veri platformunda sorguları çalıştıran farklı motorlar bulunur. Büyük veri sisteminde sorguları çalıştıran motor MapReduce, Spark [32], Tez [33] olabilir. Her bir motorun güçlü olduğu durumlar vardır. Çalışmamızda veri işleme için bu motorların güçlü olduğu alanlar değerlendirilmiş ve amaçlarına uygun olacak şekilde kullanılmışlardır.

Dosyanın tamamını okuma ve veri aktarımı yapılması durumlarında MapReduce motoru; Hive'dan sorgulama yapılması durumunda diske yazmadan hızlı sonuç getirebilen Tez motoru; akan verinin analiz edildiği durumlarda ise Spark motorunun daha güçlü olduğu tespit edilmiştir. Bu nedenle geliştirilen sistem Nifi üzerinde bileşen bazında tanımlanabilen

konfigürasyonlar sayesinde bu üç motoru da kullanabilir durumda tasarlanmıştır.

### 3.5 Veri Erişim

Büyük veri ekosisteminde bulunan verilerin yapısal farklılıkları nedeni ile bu verilere ulaşma ve bu veriler üzerinde arama yapma yöntemleri de farklılaşmaktadır. Bu amaçla üç farklı yöntemle üç ayrı servis kullanılarak HDFS'de bulunan verilere ulaşılmıştır.

**Hive**: Hive bir veri tabanı sistemi değildir. Hive'da bulunan tabloların üst verisi Hive üzerinde tutulurken; verileri HDFS dosya sisteminde tutulur. Hive, HDFS de bulunan bu yapısal verilerde SQL sorgulamalarının yapılabilmesini sağlar ve sorgulara yanıt dönerken MapReduce, Spark veya Tez motorunu kullanabilir.

**Spark**: Paralel işlem yapılmasını, bellek içerisinde veri okunması, keşif amaçlı veri analizlerinin ve makine öğrenmesi kodlarının çalıştırılmasını sağlar.

**Solr** [34]: Tam metin arama motorudur. Metinlerde bulunan sözcüklerin dizinlenmesi ve hızlı bir şekilde bu dizinlenmiş sözcükler üzerinde arama yapılabilmesi için kullanılır.

### 3.6 Veri Akışı

Büyük veri ekosistemine aktarılan verilerin özellikleri değerlendirilerek veri akışı alt yapısı oluşturulmuştur. Bu amaçla Sqoop ve Nifi kullanılmıştır.

İlişkisel veri tabanlardan Hadoop ortamına yapısal veri aktarımı yapılırken Sqoop; yapısal olmayan sosyal medya ve sunucu dosyaları gibi kaynaklardan verilerin aktarımı yapılırken ise Nifi kullanılmıştır.

### 3.7 İş Zamanlama

Nifi: Veri aktarım işlerinin zamanlayıcısı olarak Nifi' da bulunan zamanlama özelliği kullanılmıştır.

## 4. Uygulama İş Akışı

Büyük veri arşivleme sistemi uygulaması yapısal ve yapısal olmayan verilerin Hadoop ortamında arşivlenmesini sağlar. Uygulama genel iş akışı Şekil-7'de gösterilmiştir.


**Şekil-7:** Uygulama Akışı

Geliştirdiğimiz bu uygulama sayesinde son kullanıcılar büyük veri ekosistemini hakkında bilgi sahibi olmadan gerekli tüm arşivleme işlemlerini uygulama aracılığı ile yapabilmektedir.

### 4.1 Kaynak Tipi Tanımlanması

Arşivleme işleminin başlatılabilmesi için arşivlenecek kaynak tipinin sistemde tanımlanmış olması gerekir. Arşivlemenin yapılacağı kaynak tipleri MsSql, Oracle, FileServer, Twitter gibidir.

Büyük veri arşivleme yapımız Rdbms verilerinin, sunuculardaki bulunan dosyaların ve Twitter'dan tweet'lerin alınarak HDFS dizinine kaydedilmesini desteklemektedir. Yeni kaynak veri tipleri gereksinim doğrultusunda sisteme tanımlanabilir.

Her bir kaynak tipi için kaynak tiplerine uygun olacak şekilde Nifi'da yeni iş akış template'leri oluşturulur. Bu template'ler daha sonra aynı kaynak tipine ait görev tanımlamalarında kullanılır.

### 4.2 Kaynak Sunucu Tanımlanması

Hadoop sistemine aktarımımı yapılacak olan verinin bulunduğu kaynak sunucunun sistemde tanımlanması işlemidir.

Her bir kaynak tipi için gereksinim duyulan veriler farklılaşır. Kaynak sunucu tanımlanması aşamasında genel olarak sunucu bilgilerine ve sunucuya erişimi sağlayacak güvenlik bilgilerine ihtiyaç duyulur.

Örneğin, MsSql (Microsoft Sql Server) kaynak tipi için sunucu ad, sunucu iskele (port) bilgisi gereklidir. MsSql sunucuya erişim için iki seçenek bulunur; integrated security ve sql server authentication. MsSql bağlantısı için sql server authentication seçildiği taktirde kullanıcı adı ve parola bilgisi gerekecektir. Uygulama üzerinden girilen parola için Java Key Store'da JKS Alias oluşturulur ya da daha önceden oluşturulmuş bir JKS Alias bağlantı için kullanılabilir.

Oracle veri tabanına erişimlerde isetümleşik güvenlik seçeneği bulunmaz. Bağlantı için muhakkak kullanıcı adı ve parolası gerekir ve tanımlanan JKS Alias ile bağlantı sağlanabilir.

### 4.3 Görev Tanımı Yapılması

Aktarımları yapılacak yapısal, yapısal olmayan veya sosyal medya verileri için görev tanımı yapılır. Kaynak sunucu tanımlanması sırasında belirtilen kullanıcının yetkileri çarçevesinde veriler kaynak sistemlerden çekilerek Hadoop sistemine aktarılabilir.

Her görev tipine uygun iş akışları farklılaştığı gibi aynı görev tiplerinde seçilen özelliklere göre de iş akışları farklılaşabilmektedir. Önyüz katmanında görev tanımlaması yapıldığında kaynak tipi oluşturma kısmında hazırlanılmış Nifi template'ler kullanılarak Nifi'da iş akış işleri oluşturulur. Nifi'da oluşturulan işler son kullanıcıların uygulama üzerinden belirttiği zaman, tarih veya aralıklarda çalışacak olan zamanlanmış işlere dönüşür.

#### 4.3.1 Yapısal Veri Aktarımı

Yapısal verilerin aktarımı çalışmalarına başarım sınamaları ile başlanılmıştır. Aktarımı yapılacak olan yapısal veri boyutlarının büyük olması nedeni ile bu başarım sınamaları ayrıntılı biçimde yapılmıştır.

Öncelikle Nifi ile verilerin aktarımı denemeleri yapılmıştır. Nifi'ın yapısı itibari ile "streaming" aracı olması ve arka planda verilerin aktarımında birden fazla iş parçacığı oluşturmaması nedeni ile Nifi'ın veri aktarım hızı yeterli bulunmamıştır.

Diğer veri aktarım yöntemi olan Sqoop (SQL To Hadoop) ile yapılan aktarım işlerinde ise Sql ortamlardan Hadoop ortama veri aktarımının hızlı bir şekilde yapıldığı gözlemlenmiştir. Sqoop aynı anda birden fazla iş parçacığı oluşturarak Sql sistemlere atılan sorguları parçalayabilmektedir. Sqoop

kullanımı ile Hadoop ortamına yapısal verilerin toplu aktarımlarında başarım kazancı sağlanmıştır.

Sqoop ile veri aktarımı yapılırken oluşacak iş parçacıklarının sayısına son kullanıcı karar verebilir. Kullanıcının belirttiği bilgilere uygun şekilde ve eğer kullanıcı belirtmemişse Sqoop için tasarladığımız en iyi varsayılan ayarlara göre otomatik olarak Sqoop komutları oluşturulur. Oluşturulan Sqoop komutu Nifi'da zamanlanmış bir işe dönüştürülür. Diğer işlemlerde olduğu gibi bu işlemi yaparken son kullanıcının büyük veri platform bileşenleri hakkında bilgi sahibi olmasına gerek bulunmaz.

Büyük veri ortamına aktarımı yapılan verilerin kaynakta kaldığı yerden aktarımına devam edebilmesi için Sqoop'un artımlı ekleme özelliği kullanılmıştır. İşin en son nerede kaldığı bilgisi Sqoop'un kendisine ait olan bir meta veri tabanında saklanılmaktadır. Son kullanıcı, Sqoop'un artımlı ekleme yapabilmesi için takip edeceği kolon adını Şekil-8'deki gibi mapper sütunu olarak sisteme tanımlanmalıdır.

| Kaynak Kolon Adı | Şifrelenecek Kolon(lar) | Mapper | İmha |
|---|---|---|---|
| DmlLogJobHistoryId | ☐ | ☑ | ☐ |
| DmlLogTableConfigurationId | ☐ | ☐ | ☐ |
| CustomerId | ☐ | ☐ | ☐ |
| ExtensionId | ☐ | ☐ | ☐ |
| CreditCardId | ☐ | ☐ | ☐ |
| PartyId | ☐ | ☐ | ☐ |
| AuditDate | ☐ | ☐ | ☑ |
| AuditClientUserName | ☐ | ☐ | ☐ |

**Şekil-8:** Nifi Yapısal Olmayan Veri Aktarımının İş Akışı

Yapısal verinin aktarımındaki bir diğer önemli özellik ise verinin Hive'da tutulacağı formattır. Çalışma sırasında yapılan araştırmalar ve sınamalar sonrasında Orc [35] formatında verilerin sistemde tutulmasına karar verilmiştir. İyi bir Hive sıkıştırma formatı olan Parquet [36] formatı kullanıldığında Sqoop meta veri tabanında işin son kaldığı yer güncellenmediği için Parquet formatı ile artımlı ekleme yapılamamıştır. Bu nedenle Orc formatı kullanılarak Hive'da veriler saklanılmıştır. Orc formatı ile oluşturulan yapı iyi bir sıkıştırma ve query performansı da sağlamıştır.

Büyük veri ortamına aktarılan verilerin belli bir süre zarfı sonrasında yasal saklama zorunluluk süresi dolabilir. Bu nedenle uygulamanın verileri yok etme özelliği de bulunmaktadır. Bu fonksiyonel özellik sayesinde son kullanıcı dilerse HDFS'de kaydedilecek verinin yasal saklama süresi dolduğunda HDFS'den silinmesini de isteyebilir. 'Arşiv Süresi İmha (Yıl)' alanında bu verilerin ne zaman silineceği koşullarını belirtebilir. Bu özellik seçildiğinde otomatik olarak oluşturulacak yeni bir Nifi işi ile silme işlemi kullanıcının belirttiği zaman diliminde gerçekleştirilir.

### 4.3.2 Yapısal Olmayan Veri Aktarımı

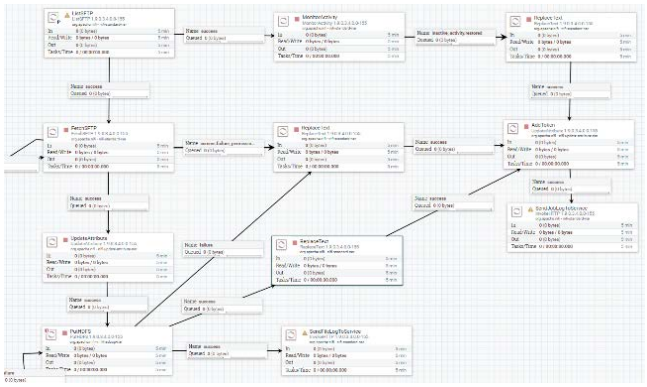
Unix sunuculardan Sftp, Windows sunuculardan ise Ftp erişimi ile yapısal olmayan verilerin Hadoop ortamına aktarımı yapılır. Yapısal veri aktarımında olduğu gibi sunucu erişim bilgileri gerekmektedir. Erişim için gerekli olan kullanıcı parola bilgisi uygulamanın veri tabanında şifreli olarak saklanılır. Uygulamanın katmanlarında ve büyük veri ortamında hiçbir şekil ve koşulda parola bilgisi açık bir şekilde yer almamaktadır.

Şekil-9'da görüldüğü üzere önyüz katmanında basit bir işlem gerçekleştirerek son kullanıcı gerekli tanımlamaları yapmış olur.


**Şekil-9:** Görev Oluşturma Ekranı – Dosya Aktarımı

Büyük veri tarafında ise Nifi'da hazırlanmış şablonlar kullanılarak önyüze göre çok daha karmaşık bir iş akışı otomatik olarak oluşturulur. Bu akış Şekil 10'da gösterilmiştir.

Yapısal verilerin aktarımında olduğu gibi yapısal olmayan verilerin aktarımında da kaynakta bulunan tüm veriler aktarılmaz ve sadece değişiklikler tespit edilerek aktarılır. Nifi veri aktarımının yapılacağı dizini dinleyerek yeni gelen veya güncellenen dosyaları tespit eder. Çalışmamızda bu amaçla zaman damgası (timestamp) yapısı kullanılmıştır.



**Şekil-10:** Nifi Yapısal Olmayan Veri Aktarımının İş Akışı

### 4.3.3 Sosyal Medya Verisi Aktarımı

Twitter'dan aktarılacak sosyal medya mesajları için mesaj ayıklama kuralları belirtilir. Bu kurallar doğrultusunda Nifi tarafında bir iş akışı oluşturulur. Sosyal medya verilerini Hadoop ortamına aktarırken içeriği boş tweetleri ayıklamak gibi veri temizliği işlemleri de yapılır. Bunun yanı sıra tweetlerin büyük veri sisteminde gereksiz yer kaplamasını önlemek için tweetler birleştirilerek sisteme kaydedilir. Birleştirme işlemi KB boyutlarındaki tweetlerin HDFS üzerinde 128 MB yer kaplamasını önlemek amacı ile yapılır.

Uygulama üzerinden yapılacak tanımlamalar ile sosyal medya verisi aktarımında tweetlerin diline, içeriğine, tweetin atıldığı konuma ve takip edilecek hesaplara göre tweetler toplanılabilir.

## 4.4 Veri Güvenliği Tanımlaması

Uygulama mevcutta var olan veri sözlüğü sistemi ile bütünleşik olarak çalışmaktadır. Aktarımı yapılacak yapısal veride kritik ya da önemli veri bulunduğunda uygulama bu alanların varlığını son kullanıcıya bildirir. Son kullanıcı yapısal veri içerisinde bulunan gizli ve önemli olan alanların seçimini yapar.

Nifi üzerinde oluşturulan iş akışları son kullanıcının şifreli aktarılması gerektiğini belirttiği alanları şifreli şekilde aktarır.

Veri içeriğinin şifrelenmesinde Homomorfik ve AES şifreleme yöntemleri kullanılmıştır.

Homomorfik şifreleme, gizliliğin sağlanması için kullanılan ve temel olarak kodlanmış verilerin hesaplanmasıyla ilgilenen bir şifreleme yöntemidir. Homomorfik özelliğe sahip şifreleme yöntemleri, veri gizliliği ve güvenliği gerektiren uygulamalar için çok uygundur [37].

AES, simetrik şifreleme standardıdır. AES'in son derece güvenli, hızlı ve güçlü bir şifreleme algoritması olduğu kanıtlanmıştır ve kullanım kolaylığı nedeni ile yaygın olarak kullanılmaktadır [38].

## 5. Büyük Veri Güvenliği

Büyük veri uygulamalarının yaygınlaşması göz önüne alındığında, giderek daha fazla veri güvenliği ve gizlilik sorunu, büyük veri uygulamalarının geliştirilmesinde büyük zorluklar getirmiştir [39].

Büyük veri tabanlı arşivleme yönetim sistemi uygulamasında hem homomorfik hem de AES tabanlı şifreleme yöntemleri kullanılmıştır.

### 5.1 Küme ve Servis Erişim Güvenliği

Büyük veri kümesine erişimlerin güvenliği için ortamda bulunan tüm hizmetler Active Directory ile bütünleştirilmiştir. Hadoop kümesi içinde bulunan hizmet kullanıcılarını ve kaynaklarını doğrulamak için büyük veri platformu uçtan uca Kerberos yapısı kullanılmıştır. Kerberos hem kullanıcı hem de hizmetler için güçlü kimlik doğrulama ve kimlik yayılımı için kullanır.

Hadoop ortamındaki veriler HDFS'de tutulmaktadır. Yapısal olan verilere Hive üzerinden; yapısal olmayan verilere Solr üzerinden sorgulama yapılabilmektedir. HDFS'de bulunan verilere erişim, Active Directory ile bütünleşmiş olarak çalışan Ranger tarafında verilmiş olan yetkiler doğrultusunda olabilmektedir. Yapılan tüm erişimler Ranger tarafından kayıt altına alınmaktadır.

### 5.2 Kaynak Sunucuya Erişim Güvenliği

Büyük veride akan verinin analizi için Nifi, verilerin toplu aktarımları için ise Sqoop kullanılmıştır.

Nifi hizmet temelli çalışan bir uygulamadır. Nifi ile Rdbms ortamlarına kullanıcı adı ve parola bilgisi gerekmeden Kerberos yöntemiyle çalışan Hadoop ortamından "Trusted Connection" bağlantı oluşturulmuştur. Nifi ile doğrudan veri aktarımı paralel veri akışına izin vermediği için Sqoop ile veri aktarımı tercih edilmiştir. Sqoop, servis tabanlı çalışmadığı için kullanıcı adı ve parola bilgisi ile kaynak sunucuya bağlantı kurabilmektedir. Sqoop ile veri aktarımlarında ihtiyaç duyulan parolalar JKS (Java Key Store) alias alt yapısı ile Linux ortamında güvenli bir şekilde saklanılır.

### 5.3 Yapısal Veri İçeriğinin Güvenliği

Verilerin sahipliği, içeriklerinin kritik, gizli veya kişisel veri içerip içermediği bilgisi kurumsal veri sözlüğünde yer alan bilgiler arasındadır. Büyük veri arşivleme uygulaması Rest API ile veri sözlüğüne erişerek aktarılacak verinin gizli veya kişisel veri içerip içermediği bilgisini veri sözlüğü uygulamasından alır. Bu bilgi büyük veri ortamına veri aktarım istağında bulunulduğunda şifrelenmesi gereken veriyi belirleme amacı ile kullanılır. Eğer aktarımı yapılacak olan veri, gizli veya kişisel veri içeriyor ise arşivleme uygulaması gerekli uyarıları son kullanıcıya bildirir. Son kullanıcı arşivleme uygulaması aracılığı ile şifrelenmesi gerekecek sütunların seçimini yaparak ilgili sütunların Hive'da şifreli olarak tutulmasını sağlar.

### 5.4 Homomorfik Şifreleme

Homomorfik şifreleme, iletişim ve depolama sürecinde verilerin gizliliğini koruma hedefine ulaşan şifrelenmiş veriler üzerinde hesaplamalara izin veren şifreleme sistemidir [40]. Bu özelliği ile Büyük Veri Arşivleme Yönetim Sistemi uygulamasında kullanımı tercih edilmiştir.

Homomorfik şifreleme yöntemi ile şifrelenmiş sayılar üzerinde yapılan matematiksel işlemlerin sonucunda oluşturulan şifreli sonucun şifresi çözüldüğünde elde edilecek sonuç ile bu işlemlerin şifresiz veriler üzerinde uygulanmasıyla elde edilecek sonuç aynı olacaktır. Bu nedenle homomorfik şifreleme yöntemi sadece sayısal veriler için kullanılabilir. Homomorfik şifreleme, temelinde kafes temelli (Lattice-Based) adı verilen matematiksel bir algoritmayı kullanır ve şifrelenmiş verinin içerisindeki şifreyi çözmeye gerek duymayan, gizlilik problemleri oluşturmayan bir şifreleme yöntemi olarak gizlilik koruma öncelikli uygulamalarda kullanılır. Aynı zamanda kriptoanalitik saldırılara karşı güvenli bir post-kuantum yöntemi olarak görülebilirler [41].

Homomorfik şifrelemenin sayısal olmayan veriler üzerinde yapılamaması nedeni ile alfa-sayısal karakter setlerinde AES şifreleme algoritması kullanılmıştır. Bu iki farklı şifreleme yapısı kullanılarak tüm veri tiplerinin şifreli olarak aktarılabildiği bir ortam oluşturulmuştur.

## 6. Sonuç

Büyük veri arşivleme ve işleme kavramları son zamanlarda üzerine çokça çalışılan kavramlar arasında yer almaktadır. Giderek artan veri analizi gereksinimlerine çözüm olarak hız, hacim ve verimlilik açısından daha etkin bir veri saklama ve işleme yöntemi gerekmektedir. Yapılan çalışmada yapısal ve yapısal olmayan veriler erişimi kolay, maliyeti düşük bir şekilde arşivlenmeye başlanmıştır.

Hadoop teknolojisi sayesinde verinin en ucuz şekilde arşivlenip depolanması, istenilen durumda en efektif şekilde veriye erişimin yapılması sağlanmıştır. Arşivlenen verinin yaşam döngüsü dinamik olarak tanımlanabilecektir ve veri gruplarına atanmış politikalar vasıtası ile yönetişimi gerçekleştirilebilecektir.

Arşivlenen verilere istenilen uygulamalardan erişim sağlanabileceği gibi, ANSI-SQL ölçünlü sorgulama diliyle de erişim yapılabilmesi için gerekli altyapı geliştirilmiştir. Bu sayede son kullanıcılar açısından veri erişiminde ve işlemesinde kullanılan ölçünler ve sorgulama dili açısından herhangi bir farklılık bulunmamaktadır.

Dinamik bir şekilde tanımlanan rol tabanlı erişim yönetimi sayesinde veri güvenliği sağlanmıştır.

Arşivlenen veriler bir katalog yapısı içerisinde tutulması, böylece arama, filtreleme gibi arşivlenen veriye rapor amaçlı ulaşım gerçekleşmiştir.

İlişkisel veri tabanı üzerinde arşivlenmiş ve mevcut sistemde 29,85 TB yer kaplamakta olan 10 adet log tablosu büyük veri

tabanlı arşivleme yönetim sistemine taşınmış, HDFS üzerindeki üçer kopyanın tüm tablolar için toplam kapladığı alan 7,7 TB olarak ölçülmüştür. Ayrıca geliştirilen veri erişim katmanı ile boyut kazancı ve yedekleme maliyetlerinin azaltılmasının yanı sıra veri sorgulama ve işlemede de %87 ila %99 oranında başarım artımı gerçekleşmiştir. Uygulanan çalışmanın mevcut hali ile giderek hızlı şekilde artan veri saklama ve işleme ihtiyacının düşük maliyet ve yüksek başarım ile sağlanmasında etkili olacağı öngörülmüştür.

Bu çalışma, TÜBİTAK Teknoloji ve Yenilik Destek Programları Başkanlığı (TEYDEB) desteği ile özel bir katılım bankası tarafından gerçekleştirilmiştir. Bu çalışma sonucunda ortaya çıkan arşivleme uygulaması üretim ortamında ilgili banka tarafından kullanılmaktadır.

Arşivlenecek olan verilerin belirlenip maliyetli veri saklama ve işleme ortamlarından büyük veri arşivleme ortamına taşınması, işlenmesi ve veri yaşam döngüsü uyarınca saklanması; verilerin formunun ve içeriğinin korunduğu daha etkin ve işlevsel bir çalışma olarak tüm paydaşlar için veri saklama ve analiz etme gereksinimlerini gidermede yardımcı olacaktır.

## Kaynakça

[1] RODADB long term digital preservation, https://www.keep.pt/en/produts/roda-long-term-digital-preservation-repository-solution/

[2] Heuscher, S., Jaermann, S., Keller-Marxer, P., Moehle, F., Providing authentic long-term archive access to complex relational data, 2004, DL/0408054, https://arxiv.org/abs/cs/0408054

[3] Brandl, S., Keller-Marxer, P., Long-term archiving of relational databases with chronos. First International Workshop on Database Preservation, March 2007, https://www.researchgate.net/publication/304374749_Long-term_Archiving_of_Relational_Databases_with_Chronos

[4] ZISMAN, A. An overview of XML. Computing & Control Engineering Journal, 2000, 11.4: 165-167.

[5] CSV, https://en.wikipedia.org/wiki/Comma-separated_values

[6] Swiss Federal Archives, https://www.bar.admin.ch/bar/en/home.html

[7] XSD XML Schema Definition, https://www.ibm.com/docs/en/iis/11.5?topic=SSZJPZ_11.5.0/com.ibm.swg.im.iis.metadata.common.doc/topics/c_xml_schema_definition_assets.html

[8] ZIP64 compression format, https://www.artpol-software.com/ZipArchive/KB/0610051629.aspx

[9] SIARD file format, https://www.loc.gov/preservation/digital/formats/fdd/fdd000426.shtml

[10] Fitzgerald, N., Using data archiving tools to preserve archival records in business systems - a case study, International Conference on Preservation of Digital Objects, 10, Lisboa, 2013, https://purl.pt/24107/1/iPres2013_PDF/Using data archiving tools to preserve archival records in business systems – a case study.pdf

[11] SIARD SUITE, https://www.bar.admin.ch/bar/en/home/archiving/tools/siard-suite.html

[12] SIARD SUITE user experiences, https://coptr.digipres.org/index.php/SIARD_Suite

[13] Preserving databases using SIARD, https://dilcis.eu/images/2020review/9_Draft_SIARD_Case_Study_1.pdf - P.10

[14] M. Farwell, Y. Fain on Angular, 13 November 2017, https://ieeexplore.ieee.org/document/8106872

[15] .netCore Entity Framework, https://docs.microsoft.com/tr-tr/ef/core/

[16] The Apache Hadoop Project, https://hadoop.apache.org/

[17] Dwivedi, K., Dubey S. K.. Analytical review on Hadoop Distributed file System, 2014, https://ieeexplore.ieee.org/abstract/document/6949336.

[18] Windows Communication Foundation (WCF) Architecture, https://docs.microsoft.com/tr-tr/dotnet/framework/wcf/architecture

[19] The Apache Nifi, https://nifi.apache.org/

[20] Aravinth S. S., Begam A. H., Shanmugapriyaa S., Sowmya S.. An efficient HADOOP frameworks SQOOP and ambari for big data processing, International Journal for Innovative Research in Science and Technology 1.10, 2015, http://www.ijirst.org/articles/IJIRSTV1I10027.pdf

[21] Master Node, https://www.geeksforgeeks.org/hadoop-cluster-properties-and-its-types/

[22] Worker Node, https://databricks.com/glossary/hadoop-cluster

[23] Hadoop cluster, https://www.simplilearn.com/what-is-a-hadoop-cluster-article

[24] Loganathan A., Sinha A., Muthuramakrishnan V. S. Natarajan. A Systematic Approach to Big Data Exploration of the Hadoop Framework, International Journal of Information & Computation Technology, 2014, ISSN 0974-2239 Volume 4, Number 9, http://ripublication.com/irph/ijict_spl/ijictv4n9spl_01.pdf.

[25] Hortonworks Data Platform, https://www.cloudera.com/products/hdp.html

[26] Dean J., Ghemawat S.. MapReduce: a flexible data processing tool, Communications of the ACM, January 2010, https://doi.org/10.1145/1629175.1629198

[27] The Apache Hive, https://hive.apache.org/

[28] Shaw S., Vermeulen A.F., Gupta A., Kjerrumgaard D.. Hive Security. In: Practical Hive. Apress, Berkeley, CA. , 2016, https://doi.org/10.1007/978-1-4842-0271-5_10

[29] Haloi S., Apache ZooKeeper Essentials, Packt Publishing Ltd, 2015

[30] Lindley A. Database Preservation Evaluation Report SIARD vs. CHRONOS, International Conference on Preservation of Digital Objects, 10, Lisboa, 2013, https://www.researchgate.net/publication/267652385_Database_Preservation_Evaluation_Report_-SIARD_vs_CHRONOS_Preserving_complex_structures_as_databases_through_a_record_centric_approach

[31] The Apache Hadoop YARN, https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[32] The Apache Spark, http://spark.apache.org/

[33] The Apache Tez, https://tez.apache.org/

[34] The Apache Solr, https://solr.apache.org

[35] The Apache Orc, https://orc.apache.org/

[36] The Apache Parquet, https://parquet.apache.org/

[37] Mohan M., Devi M. K. K., Prakash V. J.. Homomorphic encryption-state of the art, 2017, https://ieeexplore.ieee.org/document/8321774

[38] D'souza F. J., Panchal D.. Advanced encryption standard (AES) security enhancement using hybrid approach, 2017, https://ieeexplore.ieee.org/document/8229881

[39] Su C.. Big Data Security and Privacy Protection, 2019, https://ieeexplore.ieee.org/document/8920732

[40] Tourky D:, ElKawkagy M., Keshk A., Homomorphic encryption the “Holy Grail” of cryptography, 2016, https://ieeexplore.ieee.org/document/7924692

[41] Özerk Ö, Elgezen A.F., Mert A. C., Savaş E., Efficient Number Theoretic Transform Implementation on GPU forHomomorphic Encryption, February 2021, https://www.researchgate.net/publication/349075576