Güvenlik ve Gizlilik Temelinde Şirketlere Özel Bulut Depolama Çözümü (Private Cloud Storage)

Ömer Yanar

Kuveyt Türk Katılım Bankası A.Ş. Bilgi Teknolojileri Proje Yönetimi ve AR-GE Müdürlüğü
Feritpaşa Mah. Kule Cad. Kule Plaza Kat: 9 No: 9 P.K 42060 Selçuklu / KONYA
omeryanar@gmail.com

Özet. Dünya genelinde veri merkezlerinin gelişimi, internet altyapısına erişimin gün geçtikçe daha kolay ve ucuz olması, mobil cihazların gelişimi ve yaygın olarak kullanılmaya başlanması kurumları bulut altyapısına yatırım yapmaya zorlamaktadır. Öte yandan, başta bankalar olmak üzere dokümanlarının güvenliğine ve gizliliğine son derece önem veren şirketler için verilerin üçüncü kişilerle paylaşılması büyük sorun teşkil etmekte, bulut çözümlerinden uzak durmalarına yol açmaktadır. Bu bildiride, kurumların verilerini başka şirketlerle paylaşmak zorunda kalmadan, kurum bünyesinde bulunan veri merkezlerinde saklanan ve kurumun güvenlik duvarı ile korunan dosyalar için şirkete özel bir bulut sisteminin (private cloud) kurulması anlatılmaktadır. Dosyaların saklanması, yedeklenmesi ve erişilmesinde SQL Server FileStream teknolojisi kullanılmıştır. Geleneksel dosya sistemi yerine ilişkisel veri tabanı kullanımının sebepleri ve avantajlarından bahsedilmiş, dosya boyutlarının artmasıyla birlikte FileStream teknolojisinin verimliliğinin artması üzerinde durulmuştur. Ayrıca dosya boyutlarının sistem performansı üzerindeki etkisi incelenmiş, bir eşik değer belirlenerek dosyaların boyutuna göre veri tabanı tablosunda ya da FileStream üzerinde yer alması sağlanarak performans maksimize edilmeye çalışılmıştır.

Anahtar Kelimeler: Bulut Bilişim, Bulut Depolama, Cloud Computing, Cloud Storage, Private Cloud

1 Giriş

Gelişen teknoloji ile geçmişte kullanılan basılı bilgi kaynakları hızla dijital ortama taşınmaktadır. Günümüzde bilgisayarlar, insanların istedikleri dosyalara anında erişmelerini sağlayan sanal kütüphaneler haline dönüşmüş, internetin de yaygınlaşması ile dijital ortamdaki bu dosyalara birden çok kişinin zaman ve mekân bağımsız erişiminin sağlanmasına yönelik sistemler geliştirilmiştir [1].

Mevcut durum incelendiğinde kurum içinde çalışanlar işlerinde gerekli dosyaları kuruma ait ortak bir dosya sunucusunda ya da iş yerinde kullandıkları bilgisayarlarda tutmaktadır. İş yerinde barındırılan dosyalara kurum güvenlik politikaları gereği çoğu zaman dışarıdan erişim sağlanamamakta, bu dosyalara birden fazla kişinin erişimi söz konusu olduğunda ise paylaşım, yetkilendirme ve erişim sorunu ortaya çıkmaktadır. Erişilecek dosya sayısı arttığında sistemin kesintiye uğramadan verimli şekilde çalış-

ması barındırma, sınıflandırma, erişim ve yetkilendirme konularının hepsini kapsayan bir yazılım ve donanım sorunu haline gelmektedir.

Başta dosyaların her yerden ulaşılabilme ihtiyacı olmak üzere; donanım bağımlılığının azalması, dosya depolama sorunlarının azalması, verilerin kısa sürede daha fazla kişiyle paylaşılabilmesi, platform bağımlılığı olmaksızın verilere erişilebilmesi gibi gereksinimlerden dolayı dünyaca ünlü yazılım ve teknoloji firmaları tarafından dosya paylaşımı konusunda çalışmalar yapılmıştır. Bu çalışmalar sonucu ortaya çıkan uygulamalar, servis olarak sunulanlar ve uygulama olarak dağıtılanlar olmak üzere 2 başlık altında incelenebilir.

Servis olarak sunulan Dropbox, Google Drive, Apple iCloud, Microsoft OneDrive gibi uygulamalar kritik verilerin kurum dışı sunucularda tutulması gerekliliğinden veri güvenliğini tehdit etmekte, şirket içi kritik verilerin saklanması noktasında ulusal ve kurumsal riskler taşımaktadır. Amerika Birleşik Devletleri Ulusal Güvenlik Ajansı (NSA) tarafından yürütülen PRISM projesinin 2013 yılında medyaya sızdırılması ile birlikte bu endişeler iyice artmıştır [2]. Bu çözümler dosyaların bir sunucuda tutulup farklı platformlardan ulaşılabilmesini ve bu dosyaların paylaşılabilmesini sağlamakta olup bir limite kadar ücretsiz olarak sunulmakta, limit sonrası hizmetler ise ücretlendirilmektedir.

Çoğunlukla kişisel veya küçük işletmelerin kullanımı için uygun olan bu hizmetlerde, şirket organizasyon ağaçlarının bu sistemlere entegre olması; departman, rol ve kişi yönetiminin entegre bir şekilde yapılabilmesi mümkün değildir. Kapasite artırımı söz konusu olduğunda yıllık lisans ücretleri ortaya çıkmakta, var olan sistemlere entegrasyon kısıtları ve maliyetleri ile karşı karşıya kalınmaktadır [3]. Sistem sürekliliği adına hizmet vermeyi kesme riskleri dolayısı ile hizmet kesintisi riski de taşımaktadır.

Uygulama olarak dağıtılan sistemlere örnek olarak ownCloud ve SeaFile verilebilir. Bu uygulamalar açık kaynak kodlu olarak dağıtılıp kurumların kendi veri merkezlerince kullanılmasını sağlayarak güvenlik sorununu çözmekle birlikte, dosyaları her zaman dosya sisteminde tutmaları sebebiyle işlemsel bütünlüğü sağlama ve yedekleme işlemleri zorlaşmaktadır. Ayrıca bildirinin 3. bölümünde bahsedileceği gibi küçük boyutlu dosyaların (1 MB altı) dosya sisteminde tutulması, performans sorununa da yol açmaktadır [4].

Bu bildiride, doküman güvenliği ve gizliliğini ön planda tutan banka ve benzeri büyük ölçekli kurumların günlük iş hayatında kullandığı dokümanlarını, FileStream teknolojisini kullanarak kurum bünyesinde nasıl saklayabileceği anlatılmaktadır. Maliyetlerin azaltılması için dokümanların tutulacağı dosya sunucusunun her çalışma bölgesinde ayrı ayrı bulunması yerine merkezi tek bir yerde bulunması sağlanarak donanım giderlerinin azaltılmasının yanında sunucu bakımı, yedekleme vb. işlerin daha kolay yapılabilmesi hedeflenmektedir.

2 FileStream Teknolojisi

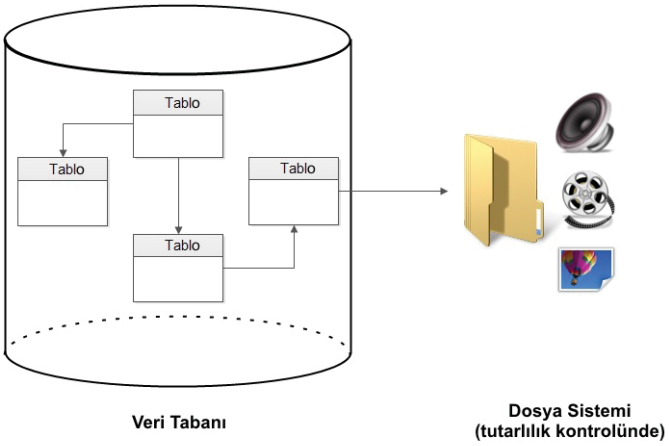
FileStream, veri tabanı olarak Microsoft SQL Server kullanan uygulamaların, doküman ve resim dosyaları gibi yapısal olmayan verileri dosya sistemi üzerinde tutmasını sağlayan teknolojidir [5]. Böylece uygulamalar, hem ilişkisel veri tabanı sistemleri

tarafından sağlanan işlemsel tutarlılık (transactional consistency), hem de dosya sistemleri tarafından sağlanan yüksek performanslı veri akışı (streaming) imkanlarına aynı anda kavuşmuş olurlar [6].

Veri depolama uygulamalarına bir yazılım sorunu olarak yaklaşıldığında, ilk ve en önemli konu, verilerin nerede tutulacağıdır. İlk akla gelen çözümler, dosyaların ilişkisel veri tabanı tabloları içerisinde ya da dosya sistemi üzerinde bulundurulmasıdır. Günümüzde birçok veri tabanı sistemi, BLOB (binary large object) olarak adlandırılan tipte veri tutulmasına imkan vermektedir ancak dosya boyutu büyüdüğünde, dosya sistemlerine kıyasla ciddi performans sorunlarına yol açmaktadır [7]. Öte yandan genel (public cloud) ya da özel (private cloud) fark etmeksizin bütün veri depolama çözümleri, kullanıcıya sistemde tutulan dosyalar hakkında birtakım ek bilgiler sunarlar. Oluşturulma tarihi, oluşturan kullanıcı, son değiştirilme tarihi gibi yapısal veriler ilişkisel veri tabanı tablolarında, asıl veriyi içeren ve yapısal olmayan dosyalar ise performans sorunundan ötürü genellikle ilişkili oldukları yapısal verilerden ayrı olarak dosya sistemlerinde tutulur. Ancak yaygın olarak kullanılan bu yaklaşım, ilişkisel veri tabanı sistemlerinin sağladığı birçok avantajın kaybedilmesine yol açmaktadır. Dosyaların veri tabanı tablosu yerine dosya sisteminde saklanması, kısaca ACID olarak adlandırılan ve veri tabanı işlemlerinin doğru ve güvenilir biçimde işleme alınarak sonlandırıldığını garanti eden sistemi sekteye uğratmaktadır [8] çünkü dosya sistemi veri tabanı işlemsel bütünlük kontrolü altında değildir.

Verilerin arşivlenmesi, yedeklenmesi ve ihtiyaç durumunda yedeklerden dönülmesi ise ayrıca ele alınması gereken bir sorundur. Verilerin tablo içerisinde tutulması halinde, veri tabanının yedeklenmesi bu sorunun çözümünü oldukça kolaylaştırmaktadır, zira yaygın olarak kullanılan birçok veri tabanı sistemi otomatik yedekleme özelliğine sahiptir. Üstelik hem yedekleme hem de yedekten geri dönme işlemi de yine yapısal bütünlük içerisinde sağlandığından, verilerin sağlıklı bir biçimde tutulduğu garanti altına alınmaktadır. Ancak yukarıda da bahsedildiği gibi, yaygın olarak kullanılan yöntemde dosyalarla ilgili yapısal veriler veri tabanında tutulurken, dosyaların kendisi dosya sistemlerinde tutulmaktadır. Dolayısıyla yedekleme işlemi, veri tabanının yanında dosya sisteminin de yedeğini almayı gerektirmekte, bu işlemler yapılırken iki sistem arasında senkronizasyonun bozulmadığından emin olunması gerekmektedir. Bu zorunluluk neticesinde ya yedekleme ve yedekten dönme işlemleri karmaşık prosedürlerden oluşmakta ya da bu işlemler esnasında sisteme erişim durdurulmaktadır.

FileStream teknolojisi, tam olarak bu soruna odaklanmaktadır ve hibrit bir çözüm sunmaktadır. FileStream ile BLOB olarak tanımlanmış bir veri tabanı kolonuna okuma yazma işlemleri, gerçekte tablo üzerinde değil dosya sistemi üzerinde gerçekleştirilmektedir. Tablo kolonunda ise, bu dosyanın yerini gösteren bir işaretçi tutulmaktadır. Öte yandan tüm okuma yazma işlemleri yine veri tabanı sunucusu tarafından yapıldığı için, işlemsel bütünlük sağlanmakta ve sistemin ACID özelliği korunmaktadır. FileStream içeren bir veri tabanı yedeklendiğinde, sistem otomatik olarak ilgili dosya sistemini de yedeklemekte, yedekten geri dönme ihtiyacı doğduğunda ekstradan senkronizasyon kontrolü ihtiyacını ortadan kaldırmaktadır. FileStream teknolojisinin yapısı, temel olarak Şekil 1'deki gibi gösterilebilir.



Şekil 1. FileStream Yapısı

Özetlemek gerekirse, FileStream tablolarında veri kolonda değil dosya sisteminde tutulur. Ancak yine de veri tabanı tarafından sağlanan işlemsel bütünlük kontrolündedir.

3 Veri Modeli

Verilerin tablo kolonunda ya da FileStream olarak dosya sisteminde tutulmasının ciddi performans farklarına yol açabileceğinden önceki bölümde bahsedilmiştir. Ancak bu performans farkı, sistemde yer alacak olan dosya boyutlarıyla yakından ilişkilidir. FileStream teknolojisinin kullanılması, veri boyutu ortalama olarak 1 MB üzerinde olduğu ve doküman okuma hızının yazma hızından daha önemli olduğu sistemlerde önerilmektedir [9]. Bu nedenle, dosya boyutlarının sistem okuma ve yazma hızını ne ölçüde etkileyeceğini görmek amacıyla farklı boyutlarda dosyalarla okuma ve yazma testleri yapılmıştır. Multi-thread bir uygulama ile her biri 4 çekirdekli olmak üzere 5 farklı istemci bilgisayardan, 16 çekirdekli bir veri tabanı sunucusuna belirlenen değişik boyutlardaki dokümanlar aynı erişim yöntemleriyle okunmuş ve yazılmıştır. Her bir istemcide 24 thread olmak üzere toplamda 120 thread çalıştırılmıştır. Testler toplam 5 milyon doküman ve 2 TB boyutunda veri içeren bir veri tabanı üzerinde yapılmış olup, veri tabanındaki kullanılabilir toplam disk alanı 5 TB boyutundadır.

3.1 Tablo kolonu üzerinde BLOB

Verilerin tablo sütunu üzerinde BLOB olarak tutulmasının sonuçları Tablo 1'de görülmektedir. Hem okuma hem de yazma hızlarının dosya boyutunun artmasıyla azaldığı gözlenmiştir. Bu sonuçlardan hareketle, büyük boyutlu yapısal olmayan verilerin veri tabanı tablolarında saklanmasının performans açısından yanlış bir tercih olacağı sonucuna varılabilir.

Dosya Boyutu (KB)	Dosya Sayısı (adet)	Okuma (MB/s)	Yazma (MB/s)
100	7180	718	605
500	1348	674	635
750	820	615	586
1000	586	586	439
1500	351	527	439
2000	235	469	371
5000	83	415	391

Tablo 1. Tablo kolonu üzerinde BLOB

3.2 FileStream Tablosu

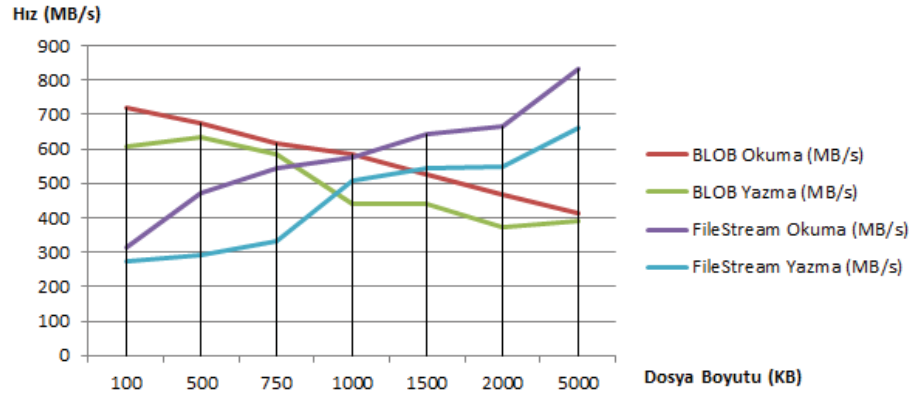
Dosyaların FileStream teknolojisi ile dosya sisteminde tutulmasının sonuçları Tablo 2'de verilmiştir. Tablo kolonunda BLOB olarak tutmanın aksine, FileStream ile okuma ve yazma performansının, dosya boyutunun artmasıyla arttığı gözlenmiştir. Ancak bu artışa rağmen FileStream 1MB altındaki dosyalarda halen daha yavaş kalmaktadır.

Dosya Boyutu (KB)	Dosya Sayısı (adet)	Okuma (MB/s)	Yazma (MB/s)
100	3130	313	273
500	948	474	293
750	723	542	330
1000	576	576	508
1500	430	645	542
2000	332	664	547
5000	166	830	659

Tablo 2. FileStream Tablosu

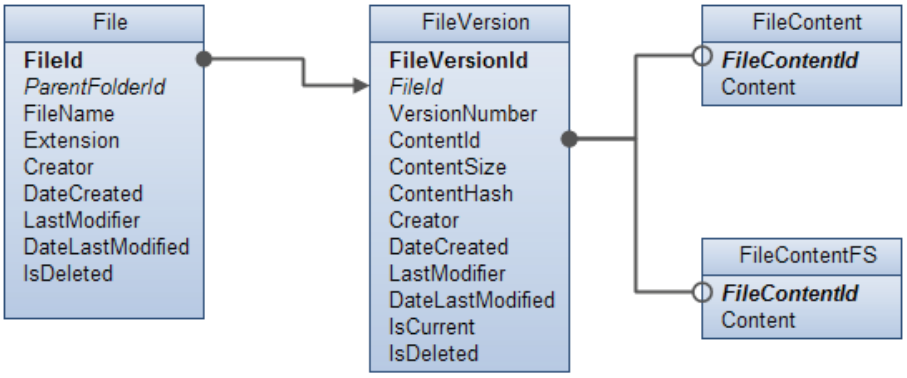
3.3 Hibrit Model

Yukarıdaki tablolardan görüleceği üzere, küçük boyutlu dosyalarda 1. yöntem, büyük boyutlu dosyalarda 2. yöntem daha uygun görünmektedir. Dosyaların kullanıcılar tarafından herhangi bir alt limit olmadan yüklenebileceği bir bulut depolama çözümünde, ortalama dosya boyutunun ne olacağını tahmin etmek çok olanaklı değildir. İşte bu yüzden, her iki sistemin de avantajını kullanmak üzere hibrit bir model kullanılması daha uygun görünmektedir. Bu modele göre dosyalar, boyutlarına bağlı olarak veri tabanında ya da dosya sisteminde tutulmaktadır. Dolayısıyla bu ayrımı yapabilmek için bir eşik değere ihtiyaç vardır. Şekil 2'de her iki yöntemin okuma ve yazma hızları karşılaştırılmaktadır. Buradan da görülebileceği gibi, 1 MB dosya boyutunda her iki yönteminde performansı yaklaşık olarak aynı olmaktadır.



Şekil 2. Okuma ve Yazma Hızlarının Dosya Boyutuna Göre Değişimi

Buradan hareketle, 1 MB eşik değer seçilerek bu değerin altındaki dosyalar doğrudan veri tabanı kolonunda, üstündeki dosyalar FileStream olarak dosya sisteminde tutulacak şekilde bir model geliştirilmiştir. Bu model, her bir dosyanın farklı versiyonları olabileceğini de göz önünde bulundurarak, bir versiyonlama sistemi de içermektedir. Geliştirdiğimiz modelin dosya, versiyon ve dosya içeriğini barındıran tablolarının tasarımı, yaklaşık olarak Şekil 3'deki gibidir.



Şekil 3. Hibrit Veri Modeli

FileContent tablosu, verileri BLOB olarak tablo kolonunda, FileContentFS tablosu ise FileStream kullanarak dosya sisteminde tutmaktadır. Eklenen her yeni versiyon, 1MB ve altında ise FileContent tablosuna, üstünde ise FileContentFS tablosuna yazılmaktadır.

4 Sonuç

Bu bildiride, güvenlik ve gizlilik kaygıları nedeniyle kurum dışı bulut depolama çözümlerinden uzak duran şirketlerin, FileStream teknolojisini kullanarak kendi ihtiyaçlarına göre geliştirebilecekleri bir özel bulut depolama (private cloud storage) çözümü üzerinde durulmuştur. FileStream teknolojisinin getirilerinden faydalanılarak, hem büyük boyutlu dosyaların dosya sisteminde tutulması sağlanmış, hem de işlemsel bütünlük (transactional consistency), yedekleme ve yedekten dönme işlemlerinin veri tabanı seviyesinde kalmasıyla karmaşık prosedürlere olan ihtiyaç ortadan kaldırılmıştır. Öte yandan, küçük boyutlu dosyalarda FileStream teknolojisinin performans sorunu ele alınmış ve hibrit bir model geliştirilmiştir.

Gerek bildiri kapsamında yapılan çalışmalar, gerekse kaynakçada belirtilen araştırmalar ve testler göstermektedir ki, 1 MB altındaki dosyalarda en iyi performans, dosyaların veri tabanı tablo kolonunda BLOB olarak tutulmasıyla elde edilmektedir. Daha büyük dosyalarda ise dosya sistemini kullanmak daha iyi sonuç vermektedir. Eğer sistem tasarımı esnasında dosyaların ortalama boyutu tahmin edilebiliyorsa, bu iki yöntemden uygun olanı tercih edilebilir. Ancak doğru bir tahminin mümkün olmadığı durumlarda, bildiride kullanılan hibrit modeli kullanmak, her iki yöntemin de avantajlarına aynı anda sahip olarak optimum verimlilik sağlayacaktır.

5 Kaynakça

1. White, C.: Consolidating, Accessing, and Analyzing Unstructured Data. Business Intelligence Network (2005)
2. Lee, T.B.: Here's everything we know about PRISM to date http://www.washingtonpost.com/blogs/wonkblog/wp/2013/06/12/heres-everything-we-know-about-prism-to-date/
3. Fabian K.: Cloud Services Review http://cloud-services-review.toptenreviews.com/
4. Azemović, J.: Varbinary vs. Filestream and other BLOB issues http://technet.microsoft.com/en-us/library/bb895234.aspx
5. Filestream Overview, http://technet.microsoft.com/tr-tr/library/bb933993(v=sql.105).aspx
6. Sebastian J., Aelterman S.: The Art of SQL Server FILESTREAM. DBA Handbooks, Simple Talk Publishing (2012)
7. Sears R., Ingen C.V., Gray J.: To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem. Technical Report, Microsoft Research, University of California at Berkeley (2006), http://research.microsoft.com/apps/pubs/?id=64525
8. Ramakrishnan R., Gehrke J.: Database Management Systems, McGraw-Hill (2002)
9. Shanmugam B.: Using SQL Server FILESTREAM to store BLOBs. White Paper, http://www.infosys.com/microsoft/resource-center/Documents/SQLServer-FILESTREAM-BLOBs.pdf