İlişkisel Veri Tabanı Sistemlerinde Çoklu Oturum ve İşlem Yönetimi NoSQL Veri Tabanı Sistemlerinde Büyük Verinin Saklanması

S. Said Aydoğan[1,2], Esat E. Demirel[1,2], Utku Ketenci[2], Mehmet S. Aktas[1],Ihsan Helvacioglu[2], Oya Kalipsiz[1]

[1] Bilgisayar Mühendisliği Bölümü, Elektrik-Elektronik Fakültesi Yıldız Teknik Üniversitesi, İstanbul
[2] Ar-Ge Merkezi, Cybersoft, İstanbul *
utku.ketenci@cs.com.tr

Kurumlar tarafından barındırılan veri ve bu verilere erişim sıklığı gittikçe artmaktadır. Artan veri boyutuyla analiz işlemleri de karmaşık hale gelmektedir. Günümüzde birden çok veri tabanı aynı amaca yönelik işlemlerde beraber kullanılabilmektedir. Veri tabanlarının ortak kullanımlarında bağlantı problemleri, hangi tablonun hangi veri tabanından geldiğinin bilinememesi, işleme (commit) ve geri alma (rollback) mekanizmalarının yeterli düzeyde sağlanamaması gibi sorunlar meydana çıkabilmektedir. Bu çalışma, birden çok veri tabanının bulunduğu sistemlerde bahsedilen problemlere bir çözüm sunmaktadır. Bu çözümle birlikte geniş ölçekte kullanılan Oracle, MySQL, MSSQL, DB2 ve Sybase gibi veri tabanları için ortak bir erişim katmanı oluşturulmaktadır. Bununla birlikte, birden çok veri tabanından elde edilen büyük hacimli verinin muhafaza edilmesi ve analiz işlemlerinde kullanılabilmesi için kolon tabanlı büyük veri işleme platformları da kullanılmaktadır. Bu platformlardan yaygın olarak kullanılan, açık-kaynaklı, HBase ve Hadoop mimarileri, büyük verinin muhafazası ve istenildiğinde erişilmesi amaçlarıyla kullanılmıştır. Gerçekleştirilen araçla farklı ilişkisel veri tabanı sistemleri ile işlemlerin (transaction) sorunsuz bir şekilde yönetilebilmesi ve bu veri tabanlarından alınan verinin büyük veri işleme platformuna taşınması sağlanabilmiştir. Geliştirilen çözümün kullanılabilirliğini ve performansını ortaya koymak adına, işlevsel ve başarım testleri gerçekleştirilmiş ve olumlu sonuçlar elde edilmiştir.

Anahtar Kelimeler: Veri Tabanı Sistemleri, İşlem Yönetimi, Büyük Veri Platformları, Dağıtık Sistemler

* Bu çalışma Yıldız Teknik Üniversitesi Yazılım Kalite Araştırma Grubu ve Cybersoft firması Ar-Ge birimlerinin işbirliği çerçevesinde gerçekleştirilmiştir. Yazarlar, Cybersoft firmasına sağlanan çalışma ortamı için; Cybersoft Ar-Ge Müdürü Umut Orçun Turgut'a ve Cybersoft çalışanlarından Şefik Temel'e de katkıları için teşekkür etmektedir. Bu çalışma aynı zamanda, Yıldız Teknik Üniversitesi BAP Projesi (Proje No: 2013-04-01-KAP03) kapsamında gerçekleştirilmiştir.

1 Giriş

Gün içinde insanlar tarafından gerçekleştirilen pek çok işlem, çeşitli veri tabanı sistemleri aracılığıyla, farklı tipteki veri kaynaklarına kaydedilmektedir. Bu durum, sürekli artan ve farklı tipteki veri tabanı sistemlerinde bulunan verinin yönetimi ve birleştirilmesi için, bir ihtiyaç olarak ortaya çıkmaktadır. Kaydedilen bu veriler ile yapılacak raporlama ve analiz çalışmalarında kullanılmak üzere, bütün bu veritabanı sistemleri ile tek bir noktadan iletişime geçebilecek bir altyapı kurulabilmektedir. Bu noktadaki zorluklar her sistemin kendine has bir altyapıya sahip olması ve bu sistemler ile iletişim kurma metotlarının farklı olmasıdır. Veri tabanlarının ortak kullanımlarında bağlantı problemleri, hangi tablonun hangi veri tabanından geldiğinin bilinememesi, işleme (commit) ve geri alma (rollback) mekanizmalarının yeterli düzeyde sağlanamaması gibi sorunlar meydana çıkabilmektedir. Bu çalışmanın amaçlarından biri, birden çok veri tabanının bulunduğu sistemlerde bahsedilen problemlere bir çözüm sunmaktadır. Bu çözümle birlikte geniş ölçekte kullanılan Oracle, MySQL, MSSQL, DB2 ve Sybase gibi veri tabanları için ortak bir erişim katmanı oluşturulmaktadır.

Birden çok veri kaynağına sahip bu sistemlerde, veri kaynağından gelen verilerin işlenerek kullanışlı bilgiye çevrilmesi gerekmektedir. Yüksek hacimli bu veriler işlenmek istendiğinde dağıtık çalışabilen NoSQL veri tabanlarında muhafaza edilebilirler. Google, ilişkisel veri tabanı yönetim sistemlerinin büyük verileri mevcut dosya sisteminde kontrol etme ve verileri efektif kullanma konusundaki sorunuyla yüzleşmiştir. Bunun neticesinde geliştirdikleri Google File System (GFS) [6], BigTable, Map/Reduce paralel işleme platformu ile sorunlara en efektif çözümleri bulurken Apache Hadoop ve Apache HBase projelerine ilham kaynağı olmuşlardır. Hadoop, Map/Reduce işlem özelliği olan Hadoop Distributed File System (HDFS) üzerine kurulmuş paralel programlama platformudur [9]. HBase ise HDFS üzerinde çalışan bir veri yönetim sistemidir [5]. Çalışmanın bir diğer amacı da, HBase NoSQL veri tabanı kullanılarak, farklı ilişkisel veri tabanlarında bulunan bilgilerin tek bir kaynakta depolanmasını sağlamaktır.

Bu bildiride; birden çok veri tabanı işletim sistemi aracılığı ile kayıt edilmiş verilerin okunarak işlenmeye hazır hale getirilmesi, farklı veri tabanı sistemlerinden gelebilecek işlemsel istisnaların (transactional exception) Spring Framework kullanılarak gerçekleştirilen işlem yöneticisi (transaction manager) aracılığı ile yönetiminin yapılması ve analizi yapılan verinin dağıtık olarak çalışan sunucularda bulunan Hadoop dağıtık dosyalama sistemine ve HBase veri tabanı sistemine taşınmasının deneyimlerinin sonuçları paylaşılmaktadır. Bu bildirinin geri kalanında sırasıyla; testlerde kullanılan ilişkisel ve ilişkisel olmayan veri tabanları ve NoSQL hakkında genel bilgiler verilecektir; bir sonraki aşamada, genel sistem mimarisi anlatılacaktır; en son bölümde de testler ve sonuçları paylaşılacaktır.

2 İlişkisel ve NoSQL Veri Tabanları

İlişkisel veri tabanı sistemleri 1960'lı yıllarda General Electric laboratuvarlarında ortaya çıkmıştır [1]. Bu tarihten önce veriler dosya yapısı içerisinde saklanırken,

ortaya çıkan bu yeni yapı ile dosyalar yerlerini ilişkisel veri tabanı tablolarına bırakmışlardır. Artan veri büyüklüğü ile ilişkisel veri tabanları ile çalışmanın zorluklarının belirgin bir şekilde ortaya çıkmaya başladığı 2010 başlarında NoSQL veri tabanları popülerlik kazanmaya başlamıştır. İlişkisel veri tabanı sistemlerine göre yüksek ölçeklenebilirlik, kümeleme, veri eşleyebilme, veri modelinin sabit olmaması ve karmaşık sorgulara karşıt olma gibi özellikleri ile [7][8] NoSQL veri tabanı sistemleri farklılık yaratmaktadır.

İlişkisel veri tabanları ACID (Atomik – Atomicity, Tutarlılık – Consistency, Yalıtım – Isolation, Süreklilik - Durability) şartlarının hepsini sağlarken, NoSQL veri tabanları bu şartları tamamen sağlamamaktadır. Bunun yerine dağıtık sistemdeki herhangi bir küme elemanında sorun olduğunda çalışmaya devam edebilmesi ve verinin bütünlüğünü koruyabilmesi şartlarını en ön sırada tutmaktadırlar. İlişkisel veri tabanlarının aksine NoSQL veri tabanlarında, verilere tekil anahtar üzerinden erişilmekte, fakat verilere erişim SQL yapısındaki kadar kolay olmamaktadır. Bu bildiri hazırlanırken popüler olarak kullanılan ilişkisel veri tabanı sistemleri ve HBase NoSQL veri tabanı sistemi incelenmiştir.

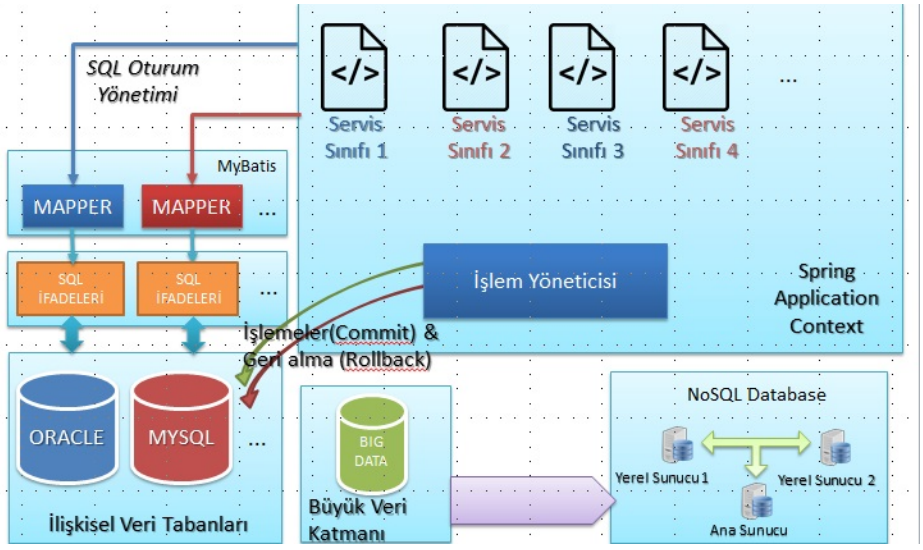
İlişkisel veritabanları arasında, Oracle, yaygın kullanılan, popüler bir veri tabanı yönetim sistemidir [2][4]. Birden çok programlama dili desteği mevcuttur [3]. MySQL ise dünya çapında en çok kullanılan açık kaynak kodlu veri tabanı yönetim sistemidir. MySQL ile kurumsal ve kurumsal olmayan binlerce uygulama geliştirilmiştir. Ayrıca Windows ve Unix/Linux işletim sistemlerini desteklemektedir [3]. Microsoft SQL Server, Windows işletim sistemine sahip bilgisayarlar üzerinde çalışabilen modern ve popüler veri tabanı sistemlerindendir. Yaygın kullanımı özellikle ".NET" uygulamaları ile olmaktadır. DB2, IBM tarafından; Sybase, SAP tarafından geliştirilen popüler veri tabanı sistemlerindendir.

HBase, kolon temelli NoSQL veri tabanı olarak sınıflandırılır. Kolon ailesi (column family) yapısı ile aynı satıra (row) ait farklı kolon aileleri oluşturularak verilere erişimin hızlandırılması mümkündür. Bir kolon ailesindeki veriye erişim, başka bir kolon ailesindeki veriye erişimi performans anlamında etkilememektedir. Başka bir deyişle kolon temelli yapı veri aktarımı esnasında satırın tamamen kilitlenmesine sebep olmamaktadır. Kolon temelli veri tabanları dışında Anahtar-Değer (Key-Value), doküman temelli ve çizge temelli gibi NoSQL veri tabanı tipleri de bulunmaktadır. HBase diğer NoSQL veri tabanı sistemlerinden, HDFS alt yapısını kullanması ile fark yaratmaktadır. Kendine ait ve tam performansla çalışabileceği dağıtık bir dosyalama sisteminin varlığı HBase'i elastik ve ölçeklenebilir kılmaktadır. Bununla birlikte, rastlantısal okuma ve yazma konusunda etkili olması HBase'in diğer NoSQL veri tabanı sistemlerine göre avantaj sağladığı başka bir noktadır. Facebook Messages'ın da tercih ettiği ve yaptıkları testlerde en iyi sonuçları veren veri tabanı sistemi NoSQL olmuştur [8].

3 Sistem Mimarisi

Gerçekleştirilen çalışma esnasında, farklı veri tabanlarına bağlanmak için Spring Çatısı (Framework) ve MyBatis Sürerlik Çatısı (Persistence Framework) kullanılmıştır. Genel sistem mimarisi Şekil 1'de gösterilmektedir.

Spring, bir uygulamanın gerçekleştirilmesi esnasında gerekecek birçok modülü içinde barındıran bir uygulama çatısıdır. Bu çalışmada Spring'in işlem yöneticisi (Transaction Manager) modülüne yer verilmiştir.



Şekil 1: Genel Sistem Tasarımı .

MyBatis Sürerlik Çatısı, ilişki nesne modellerinden (ORM) farklı olarak saklı yordamların (stored procedure) nesne olarak Java ortamında tutulmasına veya SQL cümlelerinin birer Java metotu olarak kullanılmasına olanak vermektedir. MyBatis, kullanıldığı projelerde, genel olarak, tek bir veri tabanı kaynak olarak kullanılmaktadır.

Spring Application-Context modülü işlem yöneticisini barındırmaktadır. MyBatis modülü, eşleyicileri (mapper) tutmaktadır ve SQL ifadeleri aracılığı ile veri tabanındaki verinin çekilmesini sağlamaktadır. İşlem yöneticisi SQL cümleciklerinin çalıştırılması esnasında doğabilecek istisnalar doğrultusunda işlemleri (Commit ve Rollback) yönetmektedir

Çalışma prensibi aşağıda maddeler halinde verilmektedir:

1. Kullanılacak veri tabanı sisteminin bilgileri (ör: Veri tabanı tipi, IP değeri , port, uid, parola, vb.) ile SqlSessionFactory (MyBatis sınıfı) oluşturulmaktadır.
2. MyBatis'te veri tabanı işlemleri (ör: create, update, insert, vb.) servisler aracılığı ile gerçekleştirilmektedir. Bu servisler veri tabanı yapısındaki tablo modellerine uygun şekilde çalışmaktadırlar. Servislerle modeller arasındaki etkileşim eşleyiciler (mapper) aracılığı ile sağlanmaktadır. Servislerin çalışabilmesi için bir önceki aşamada yaratılan SqlSessionFactory kullanılarak veri tabanı oturumu oluşturulmaktadır.

3. Bu aşamada istemci, eşleyicilerde tanımlanan fonksiyonlar aracılığıyla veri tabanındaki modele erişimini sağlayabilmektedir.
4. Veri tabanında istenilen değişiklik veya görüntüleme yapıldıktan sonra, istemci tarafındaki veri tabanı oturumu yine istemci tarafından kapatılmaktadır.

Bu çalışmanın hedeflediği gereksinimlerinden biri olan farklı veri tabanlarına bağlanma ve işlem yönetimi Bölüm 3.1'de, verilerin HBase'e aktarımı ve erişimi Bölüm 3.2'de anlatılmaktadır.

3.1 Farklı İlişkisel Veri Tabanı Sistemlerine Bağlanabilme ve İşlemlerin Yönetilmesi

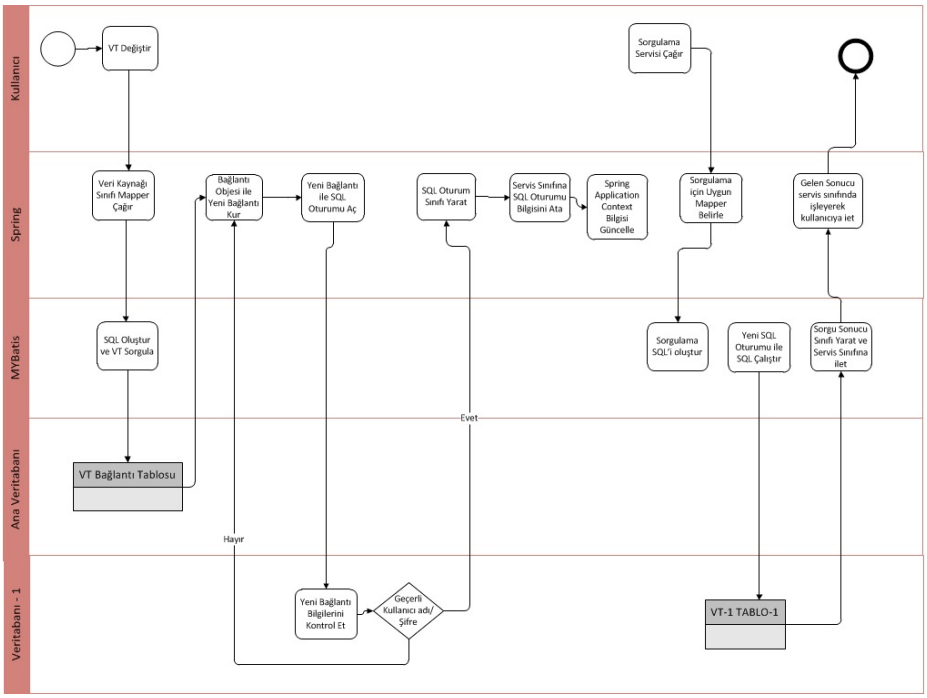
Bu projede Oracle, MSSQL, MySQL, Sybase ve DB2 veri tabanı sistemlerinin hepsi birden veri kaynağı olarak kullanmak istenmektedir. Bunun bir sonucu olarak yukarıda anlatılan MyBatis çalışma prensibi Spring'in özellikleri kullanılarak geliştirilmiştir. MyBatis-Spring Kütüphanesi aracılığı ile Spring'in MyBatis ile entegrasyonu gerçekleştirilmiştir (Bknz. Şekil 2). Entegrasyonun sonucunda ortaya çıkan yapı şu şekilde çalışmaktadır:

1. Spring, Application-Context Container modülü içinde servis yönetimini gerçekleştirir. Bu modül içerisindeki işlevsel nesnelerden biri olan "Datasource Bean", proje kapsamında kullanılacak farklı veri tabanı sistemlerine ait bilgilere erişimi sağlamaktadır (XML, Veri tabanı, Metin Belgesi veya Hard coded şekilde verilmiş olan). "Datasource Bean", bu bilgileri kullanarak, işlem yapılacak veri tabanı sisteminde bağlantı (connection) açmaktadır.
2. Bir diğer işlevsel nesne olan "SqlSession Bean", "Datasource Bean"'in oluşturduğu bağlantıyı kullanarak veri tabanı oturumunu açmaktadır.
3. Veri tabanı üzerindeki işlemleri gerçekleştirmek amacıyla kullanılan MyBatis servisleri birer "Bean" nesnesi olarak Container içerisinde yazılımcı tarafından tanımlanmaktadır. Bu servisler oluşturulurken bir önceki adımda yaratılan SqlSession enjekte edilmektedir. Bağımlılık enjeksiyonu (Dependency Injection) diye de adlandırılan bu yapı sayesinde enjekte edilen oturum bilgisine göre servis, farklı veri tabanı sistemi ile etkileşebilmektedir.
4. Aktif olan veri tabanında işlem yaptıktan sonra başka bir veri tabanıyla etkileşime geçilmek istenebilir. Bu durumda, "Datasource Bean"'inin kullandığı aktif veri tabanı bilgileri bu veri tabanının bilgileri ile değiştirilir. Bu değişim bir metot ya da arayüzden gelen bir istek ile gerçekleştirilebilir. Değişim tamamlandıktan sonra Container yenilenir. Servisler yeni oturum bilgileri ile güncellenir ve etkileşime hazır hale getirilir.

Farklı veri tabanlarına bağlantı ve sorgulama amacı ile kurulan sistem aşağıdaki sırada işlemleri gerçekleştirmektedir.

1. Veri tabanı değişikliği istemi gönderilir.
2. Bağlanılacak veri tabanına ait bilgiler Spring ApplicationContext Container'a iletilir.

3. Bağlantı Açılır.
4. "SqlSession Bean" bağlantıyı sağlar.
5. Veri tabanında oturum açılır.
6. Oturum "Service Bean"'e enjekte edilir.
7. Veri tabanındaki işlemi gerçekleştirilecek servis çağırılır.
8. Servis, Mapper'ın gerekli metodunu çağırır.
9. Mapper çağırılan metota uygun SQL cümleciğini bulur.
10. Başlangıçta karar verilmiş olan veri tabanı ile etkileşime geçerek SQL cümleciği koşturulur.
11. Sorgu sonucu "Service Bean"'e ulaştırılır.
12. Kullanıcı tarafına servis çağrısının cevabı ulaştırılır.



Şekil 2: Çoklu Veri tabanı Erişimi

Farklı veri tabanları ile çalışırken karşılaşılan bir sıkıntı da istisnaların (exception) yönetimidir. Bu noktada da Spring Framework'ün sağladığı işlem yönetimi (Transaction Management) modülü kullanılmaktadır. Servislerin içerisindeki metotlardan işlemsel olanlar (Transactional notasyonuna sahip olanlar) istisna ile karşılaşılan durumlarda geri alma (Rollback) yapabilme özelliğine sahip olurlar. Detaylı anlatımı örnek üzerinde Bölüm 4.1.'de yapılacaktır.

Veri tabanları üzerinde kullanıcı tarafından yaratılacak olan sorgular için ilgili şemada hangi tabloların bulunduğu bilgisine ihtiyaç duyulmaktadır. Her veri

tabanının farklı bir yapıya sahip olmasından dolayı bu tabloları listeleyebilmek probleme neden olmaktadır. Projede bu probleme çözüm için Java'nın veri tabanı tablo isim listesi, view isim listesi gibi meta data bilgilerini getiren java.sql paketi içindeki DatabaseMetaData sınıfı kullanılmaktadır. Bu sınıf birçok veri tabanı tasarımcısı kuruluş ile ortak oluşturulmuştur ve veri tabanı ile ilgili yapısal bütün bilgileri bize sunabilmektedir.

3.2 Büyük Veri Tabanı Sistemlerine Veri Aktarımı

HBase, Hadoop ile en verimli şekilde çalışmaktadır. Hadoop'un dağıtık mimarisi HBase'in performans ve sağlamlık açısından optimum düzeyde çalışmasını sağlamaktadır. Bu sebepten çalışmamızda ilk aşama olarak Hadoop kurulumu gerçekleşmiştir.

Bu aşamada Hadoop'un kurulacak olan versiyonunu seçerken, çalışılmaya karar verilmiş olan HBase versiyonu ile uyumluluğuna dikkat edilmiştir. Gerçekleştirilen çalışmada HBase 0.98.9-Hadoop2 versiyonu ve buna uyumlu olan Hadoop 2.5.2 kurulmuştur. Kurulumumuzda 3 tane işçi (slave) ve 1 tane üstat (master) düğümleri bulunmaktadır. Kurulum sırasında, makinalar Centos 6.5 işletim sistemine sahiptir.

Bu makinaları hepsinin üstüne Hadoop kurulumu gerçekleştirilirken işletim sistemine ve Hadoop'a ait "hosts", "core-site.xml" ve "hdfs-site.xml" dosyaları üzerinde ayarların yapılması gerekmektedir. Bu dosyalar Hadoop'un temel 3 süreci (process) olan Namenode (veri ağacı yapısını saklar cluster içinde hangi verinin nerede tutulduğunun kaydını barındırır), Datanode (verileri saklar) ve SecondaryNamenode'un (Namenode'un yedeğidir) çalışmasında temel olmaktadır. Bir master makinada Namenode tutulurken işçi makinalarda Datanode çalışmaktadır.

SecondaryNamenode, Namenode'dan başka bir makinada tutulmaktadır. Bu sayede Namenode'un çökmesi durumunda SecondaryNamenode'daki bilgiler kullanılarak sistem ayağa kaldırılabilir. "Hosts", "core-site.xml" ve "hdfs-site.xml" dosyaları master ve slave makinaların her biri için ayarlanmalıdır.

Hadoop kurulumu ve konfigürasyonu tamamlandıktan sonra HBase kurulumu gerçekleştirilir. HBase'in temel süreçleri HMaster (Namenode'un koşturulduğu makinada koşturulur metadataları saklar.), HRegionServer (Datanode'ların çalıştırıldığı makinalarda çalıştırılır, verilerin tablo yapısı altında saklanmasını sağlar.) ve HQuorumPeer (Hmaster ve Hregionserverlar arasında bilgi alışverişinin yapılmasını sağlar. İstemci tarafından gelen istekleri yönetir.)'dir.

HBase kurulurken "hbase-site.xml" konfigüre edilmektedir. HDFS'nin yordamı (HDFS Path) burada belirtilmektedir. Yapılan konfigürasyon HBase'in tablolarını HDFS yapısı altında tutmasını sağlamaktadır. Ayrıca, Hadoop'un çalışma moduna göre (Fully Distributed, Pseudo Distributed, Single Node) HBase'in çalışma modu bu dosyada belirlenmektedir. Bu çalışmada single node (tek makina) ve fully distributed (tamamen dağıtık) modlar testlerde kullanılmaktadır.

HBase'in tablo yapısı kolon aileleri (Column Family) içermektedir. Ayrıca, veriyi baytlar halinde tuttuğundan dolayı kayıt edilen verinin baytlara çevirilerek kayıt edilmesi gerekmektedir. JSON kaydedilecek verinin sınıf yapısında

baytlara çevrilerek kaydedilmesini sağlamaktadır. Yapılacak testlerde kolon aileleri kullanılarak veri kayıt edilmesi ve verinin JSON objesine çevrilerek kayıt edilmesi ayrı ayrı incelenmektedir. Bir sonraki bölümde gerçekleştirilen testler incelenecek ve sonuçlar paylaşılacaktır.

4 Gerçekleştirilen Testler ve Sonuçları

Sistem gereksinimleri dahilinde 5 farklı veri tabanı işletim sistemine erişim performans testleri ve işlem yöneticisi fonksiyonalite testleri gerçekleştirilmiştir. Bunlarla birlikte HBase veri tabanı üzerinde büyük veri yazma ve okuma testleri yapılmıştır. Bölüm 4.1'de ilişkisel veri tabanları üzerinde yapılan testler, Bölüm 4.2'de HBase üzerinde gerçekleştirilen testler detaylandırılacaktır.

4.1 Çoklu ilişkisel veri tabanı sistemlerine erişim testleri

İlişkisel veri tabanı sistemlerine erişim testleri 3.4 GHz i7 işlemciye, 8Gb RAM'e sahip bir makina üzerinde gerçekleştirilmiştir. İlişkisel veri tabanına yönelik fonksiyonel testler olarak, bağlantı kontrolü ve işlem yönetimi testleri gerçekleştirilmiştir. Bağlantı kontrolünde (Şekil 2'de detaylandırılan), veri tabanlarına bağlantı ve oturum açma kontrolü bahsi geçen tüm veri tabanları için gerçekleştirilmiştir. Bu test sırasında veri tabanlarından sadece veri getirme amacıyla kullanılan (select sorguları içeren) servisler çağrılmıştır. Servis çağrılarının sonucunda her veri tabanının kendisine ait olan veriyi eksiksiz bir şekilde getirdiği görülmüştür.

Farklı veri tabanı işletim sistemlerinde bulunan, index içermeyen ve 8 kolonluk "varchar" bilgi saklayan tablolardan 5000 satırın getirilmesi (select) ile gerçekleştirilen testlerin sonuçları Tablo 1'de gösterilmektedir. MyBatis-Spring kütüphanesi ile geliştirilen ilişkisel veri tabanı erişimi testlerinin sonuçları Tablo 1'in ilk satırında, JDBC ile gerçekleştirilen testlerin sonuçları tablonun ikinci satırında gösterilmektedir. Test sonuçlarına göre MyBatis-Spring kütüphanesi ile oluşturulan sistemin sonuçları JDBC bağlantısına göre her veri tabanı için daha iyi sonuçlar vermektedir.

Tablo 1: 5000 satırın okunması (Saniye)

	MySQL	MSSQL	ORACLE	SYBASE	DB2
JDBC	2,01	2,06	2,14	2,32	2,45
MyBatis-Spring	0,55	1,19	2,12	1,50	2,36

İşlem yönetimi testinde, Cybersoft tarafından kullanılan hatasız çalışması beklenen servisler kullanılmaktadır. Örneğin bazı durumlarda, kullanıcının web uygulaması üzerinde oturum açıp, işlediği veriyi kaydetmek için ilgili servisi çağırması gerekmektedir. Şekil 1'de gösterildiği üzere Application Context içerisinde bulunan ilgili servis, aktif olan veri tabanı oturumu kullanılarak çalıştırılır.

Böylelikle, kayıt işlemlerini kendisine enjekte edilmiş oturum üzerinden gerçekleştirebilmektedir.

Eğer metot işlemselse yani "@transactional" notasyonuna sahip ise; Application Context içerisinde yer alan işlem yönetimi (transaction manager) aracılığıyla istisnaları yönetebilmektedir. Metot içerisinde istisna (exception) ortaya çıkarsa yapılan işlemin geri alınmasını (rollback) ya da istisna oluşmamışsa metottan dönüldüğü sırada işlemin onaylanmasının (commit) sağlaması gerçekleşmektedir. Bahsedilen örnek için kayıt sırasında istisna çıkaran durum ve çıkarmayan durum ayrı ayrı test edilmiştir.

İşlem yönetimi fonksiyonalite testleri sonucunda, istisna meydana geldiğinde kaydetme işleminin gerçekleşmediği, istisna olmadığı durumda ise kullanıcının belirttiği veri tabanı üzerinde kayıtların oluştuğu görülmüştür.

İşlem yönetiminin zamansal olarak yükünü gözlemlemek amacıyla gerçekleştirilen performans testleri sonucu Tablo 2'de sergilenmektedir. Ekleme işleminde kullanılan veri 8 kolonlu varchar içeren 1000 satırdan oluşmaktadır. İlişkisel veri tabanlarına eklenmesi önceden yaratılmış üzerinde index bulunmayan bir tablo yapısı kullanılarak gerçekleştirilmiştir. Tablo 2'de ilk satır "@transactional" noyasyonuna sahip, ikinci satır ise sahip olmayan metotların sonucunu göstermektedir.

Görüleceği üzere işlemsellik düşük seviyede fazla yük yaratmaktadır. Bununla birlikte, sebep olduğu küçük dezavantaja rağmen sistemi hataya karşı daha sağlam kılan (commit ve rollback) mekanizmaları içerdiğinden ötürü kullanılması fayda sağlamaktadır.

Tablo 2: 1000 satırın eklenmesi (Saniye)

	MySQL	MSSQL	ORACLE	SYBASE	DB2
İşlemsel	14	16	55	15	17
İşlemsel değil	10	13	54	10	14

4.2 HBase veri tabanı performans testleri

HBase ve Hadoop, tek makina ve tam dağıtık şekilde Cybersoft'a ait sunucular üzerinde gerçekleştirilmiştir. Tek makina tipindeki kurulum 2GB RAM, 3 GHz tek çekirdek işlemciye sahip bir sunucuda çalıştırılmıştır. Tam dağıtık modda, master makina 2 GB Ram ve 3 GHz çift çekirdeğe sahip iken, slave makinalar her biri 1 gb RAM ve 3GHz tek çekirdek işlemciye sahip 2 makina üzerinde kurulmuştur.

HBase yapısı altında verilerin eklenmesi ve okunması zamansal etki açısından incelenmiştir. Aşağıda sonuçları sunulan testler, UCI Yapay Öğrenme Veri kaynağından (http://archive.ics.uci.edu/ml/datasets/Poker+Hand) çekilen 1025010 satır ve 11 kolon içeren veri kümesi ile gerçekleştirilmiştir.

Tablo 3'te tek makina ve tam dağıtık konfigürasyonları üzerinde sıralı (synchronous) ve toplu (batch) veri ekleme sonuçları gösterilmektedir. Toplu olarak veri ekleme sonuçları beklendiği gibi daha iyi sonuç vermektedir (30 kat daha iyi). Tam dağıtık yapılandırmada ise toplu veri ekleme sonuçları beklenildiği gibi tek makinadan daha iyi sonuç vermektedir.

Tablo 3: 1 milyon satırın eklenmesi (Saniye)

	Senkron	Toplu
Tek Makina	1865	66
Tam dağıtık	2129	26

Tablo 4'te tek makina ve tam dağıtık konfigürasyonları üzerinde JSON'a çevirilerek ve kolon ailelerine bölünerek veri ekleme sonuçları gösterilmektedir. Sınıf yapısına sahip JSON objeleri ile tek bir kolon ailesi üzerinden yapılan testlerin çoklu kolon ailesi yapısı ile gerçekleştirilen testlere göre daha iyi sonuç verdiği saptanmıştır.

Tablo 4: 1 milyon satırın farklı yapılarda eklenmesi (Saniye)

	JSON	Kolon Ailesi
Tek Makina	37	111
Tam dağıtık	25	97

Tablo 5'te tek makina ve tam dağıtık yapılandırmaları üzerinde okuma işlemlerinin sonuçları gösterilmektedir. HBase yapısı ile tek makina (3 GHz işlemci ve 2 GB RAM) üzerinde yapılan testlerin sonucunda 1025010 satırın okunması 44 saniye sürmüştür. Şu ana kadar gerçekleştirdiğimiz testlerde ilişkisel veri tabanı sistemlerinden bu seviyede bir veri okuması gerçekleştirilememiştir (3.4 GHz i7 işlemciye, 8Gb RAM'e sahip bir makina üzerinde). Basit bir hesapla 5000 satırlık verinin okunması en iyi ihtimalle (MySQL) 0,55 sn sürdüğüne göre 1 milyon satırlık verinin okunması 0,55*200=110 sn sürecektir.

Tablo 5: 1 milyon satırın okunması (Saniye)

	Okuma
Tek Makina	44
Tam dağıtık	45

Elde edilen sonuçlara dayalı olarak, HBase' in 3 kat daha hızlı veri döndüğü saptanmıştır. Veri okunması kısmında tam dağıtık yapılandırmada verinin her bir makinadan toplanarak getirilmesi söz konusu olduğundan dolayı tek makine ve tam dağıtık yapılandırma arasında belirgin bir süre farkına rastlanmamıştır.

5 Sonuç ve Gelecekteki Çalışmalar

Bu çalışma kapsamında, birden fazla veri tabanının ortak kullanımının olduğu sistemlerde karşılaşılan problemlere bir çözüm geliştirilmektedir. Bu problemler arasında, bağlantı problemleri, hangi tablonun hangi veri tabanından geldiğinin bilinememesi, işleme ve geri alma mekanizmalarının yeterli düzeyde sağlanamaması yer almaktadır.

Önerilen çözüm, Oracle, MySQL, MSSQL, DB2 ve Sybase gibi veri tabanları için ortak bir erişim katmanı oluşturmakta ve bu farklı veri tabanlarından elde edilen büyük hacimli verinin muhafaza edilmesi ve analiz işlemlerinde kullanılabilmesi için kolon tabanlı büyük veri işleme platformlarını kullanmaktadır. Çözümün kullanılabilirliği test etmek amacıyla, 5 farklı veri tabanına erişim performans testleri ve işlem yöneticisi fonksiyonalite testleri gerçekleştirilmiştir. Bunlarla birlikte HBase veri tabanı üzerinde büyük veri yazma ve okuma testleri de yapılmıştır. Testler sonucunda elde edilen sonuçlar, önerilen yaklaşımın, işlemsellik açısında düşük seviyede ek yük yarattığını göstermektedir. Bununla birlikte, sebep olduğu küçük dezavantaja rağmen sistemi hataya karşı daha sağlam kılan commit ve rollback mekanizmaları sayesinde faydalar sağlamaktadır. Yine elde edilen, test sonuçlarına göre, çözümde kullanılan MyBatis ve Spring kütüphaneleri ile oluşturulan sistemin sonuçları JDBC bağlantısına göre her veri tabanı için daha iyi sonuçlar vermektedir. Toplu olarak veri ekleme sonuçları ise beklendiği gibi daha iyi sonuç vermiştir.

Gelecek çalışmalar arasında, önerdiğimiz çözümün daha farklı veri tabanları üzerinde test edilmesi ve daha kapsamlı performans ve yük testlerinin yapılması yer almaktadır.

Kaynaklar

1. Database management systems. http://db-engines.com/en/systems, note= Ulaşım: 09.01.2015, key= 0
2. Db-engines ranking. http://db-engines.com/en/ranking, note= Ulaşım: 09.01.2015, key= 1
3. Oracle system properties. http://db-engines.com/en/system/Oracle, note= Ulaşım: 09.01.2015, key= 2
4. Alapati, S.R.: Expert Oracle Database 11G Administration. Apress, Berkely, CA, USA, new edn. (2008)
5. George, L.: HBase: the definitive guide. " O'Reilly Media, Inc." (2011)
6. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: ACM SIGOPS operating systems review. vol. 37, pp. 29–43. ACM (2003)
7. Leavitt, N.: Will nosql databases live up to their promise? Computer 43(2), 12–14 (2010)

8. Muthukkaruppan, K.: Storage infrastructure behind facebook messages. Proceedings of HPTS 11 (2011)
9. White, T.: Hadoop: The definitive guide. " O'Reilly Media, Inc." (2012)