Veri Büyüklüklerinin Veritabanı Yönetim Sistemlerinde Meydana Getirdiği Değişim: NoSQL

Yılmaz GÖKŞEN, Hakan AŞAN

Dokuz Eylül Üniversitesi İktisadi ve İdari Bilimler Fakültesi Yönetim Bilişim Sistemleri Bölümü
yilmaz.goksen@deu.edu.tr, hakan.asan@hotmail.com
(Geliş/Received: 15.04.2015; Kabul/Accepted: 21.07.2015)
DOI: 10.17671/btd.52374

Özet- Bilgisayar ve iletişim teknolojilerinin gelişimi her geçen gün daha fazla organizasyonun süreçlerini ve iş modellerini değiştirmektedir. Bu hızlı değişim, bireysel ve kurumsal açılardan, çok büyük verilerin üretildiği, bu verilerin saklanmasının, işlenmesinin ve yönetilmesinin çok önemli olduğu bir gerekliliğe yol açmaktadır. Bu bilgiler ışığında yoğun okuma yazma gibi veritabanı işlemleri yapan organizasyonlar, performans ve esneklik gibi çeşitli ölçütler nedeniyle ilişkisel veritabanı yönetim sistemleri yerine farklı veritabanı tasarımları üzerinde durmaktadırlar. Bu veritabanı sistemlerinden bir tanesi de NoSQL adını verdiğimiz veritabanlarıdır. NoSQL sağladığı özelliklerle, ortaya çıkan boşluğu doldurmuş ve uygulamada farkındalık yaratmıştır. Amazon, eBay, Linkedin vb e-ticaret ve e-iş uygulayan organizasyonlar, büyüyen verileri saklamak ve farklı hizmetlerinin doğru sunumu için NoSQL' e güvenmektedir.

NoSQL veritabanları, bazı organizasyonlar tarafından kendi bünyelerinde geliştirilse bile, yaygın olarak kullanılan NoSQL veritabanları da bulunmaktadır. Bazı organizasyonlar NoSQL veritabanlarını hibrit bir yapıda ilişkisel veritabanları ile birlikte kullanmaktadır. NoSQL; yatay ölçeklendirebilme, hız, dağıtık sistemlere uygunluk, verimlilik vb. özellikler açısından fark yaratsa da, birden fazla tablo üzerinde birlikte işlem yapıldığı durumlarda sorun oluşturabilmektedir. Vurgulanan farklı yapıların özellikleri açısından, bu çalışmada, NoSQL veritabanları ile ilişkisel veritabanları karşılaştırılmalı olarak incelenmiş ve politika önerileri ortaya konulmaya çalışılmıştır.

Anahtar Kelimeler- İlişkisel Veritabanı, Veritabanı Yönetim Sistemleri, NoSQL

Induced Change of Data Size in Database Management Systems: NoSQL

ABSTRACT-Computer and communication technologies are changing the processes of more organizations and their business models every day. This fast change causes a formation in which a very large amount of data are produced and the storage, processing and managing of these data are very important from the individual and institutional aspects. In the light of this information, organizations engaged in intensive database operations such as reading and writing, dwell on new structures apart from relational database because of the criteria such as performance and flexibility. In such a process NoSQL, with the qualifications it has, has filled the gap and day by day its importance and awareness are increasing in practice. E-commerce and e-business organizations like Amazon, E-Bay and Linkedin rely on NoSQL for their growing data and accurate representation of their different services.

Even if NoSQL is generally developed by various organizations in-house, it is being used in a hybrid structure with relational databases in many organizations. Although NoSQL makes a difference from the point of horizontal scalability, speed, suitability to distributed systems, efficiency and the like, in cases where the operation is performed on multiple tables, it may create a problem because it doesn't allow cascading operations. In this study, NoSQL databases and relational databases have been examined in terms of highlighted different features and policy recommendations have been tried to be put forward.

Keywords-Relational Database, Database Management Systems, NoSQL

1. GİRİŞ (INTRODUCTION)

Enformasyon toplumunun meyvelerini, her cepte bir mobil telefon, her sırt çantasında bir bilgisayar, her ofiste bir büyük enformasyon teknolojisi sistemi olarak her yerde görmek kolaydır fakat burada en az dikkat edilen enformasyonun kendisidir. Yarım yüzyıl önce bilgisayarların toplum tarafından kullanımına başlanmasıyla ve her geçen gün oluşan yenilikler ile birlikte çeşitli veriler yığılmaya başlamıştır. Dünya sadece daha çok enformasyonla kaplanmakla kalmamış, ayrıca enformasyon da giderek büyümüştür. Veri ölçeğindeki değişim mevcut durumu da değiştirmiştir. İlk olarak 2000'ler enformasyon patlamasını deneyimleyen astronomi ve genetik gibi bilim dalları, bu patlamayı big data kavramı olarak isimlendirmiştir. Bu kavram günümüzde, insan uğraşı alanlarının tümüne yayılmıştır [1].

Bu kadar büyük ve yeni yeni içselleştirilen verilerin depolanması, işlenmesi ve karar vericilere raporlanması bilinen yöntemlerin ötesinde bir teknik ve donanım çözümü gerektirmektedir.

Başlangıçta bilgisayar sistemleri tek işlem yapmaktadır yani bir dizi içinde baştan sona tek bir işlem yapılmaktadır. Daha sonraları makinede birden fazla iş yapılmaya başlanmış ve bu da ona çoklu işlem yapma becerisi kazandırmıştır. Hâlâ işler birbirinden bağımsızdır ve işlemlerin yapılması donanımsal açıdan beklemeyi gerektirmektedir. Bu durum sonucunda istemci-sunucu mimarisi olarak SQL (Structured Query Language) veritabanları kullanılmaya başlanmıştır [2].

Son birkaç yılda veritabanı yönetim sistemi üreticileri kendi sorgulama dilleri üzerine yoğunlaşmışlardır. NoSQL fikri ile birlikte bu değişim sadece veritabanı yönetim sistemleri üreticilerini değil aynı zamanda Amazon, Facebook, Google ve Yahoo gibi ünlü Web 2.0 firmalarını da etkilemiştir [3]. NoSQL, klasik olarak bilinen ve kullanılan ilişkisel veritabanı yönetim sistemlerinden (RDBMS-Relational Database Management Systems) farklı olan, alışılmışın dışında bir yapıdır. Tasarımcısı olan Carlo Strozzi, NoSQL hareketi "ilişkisel modelden tamamen ayrılırken ona bundan dolayı daha uygun olarak ilişkisel olmayan anlamında 'NoREL' (No RELation) veya benzeri bir ad vermek gerektiğini ileri sürmektedir".

2. NOSQL MİMARİSİ (NOSQL ARCHİTECTURE)

Tipik modern ilişkisel veri tabanları; çok sayıda belgeyi içerik olarak sıralayıp, yoğun trafiği olan Web sayfalarında etkin performans göstermişlerdir. Tipik İlişkisel VeriTabanı Yönetim Sistemi (İVTYS) uygulamaları ya küçük fakat sıkça oku/yaz işlemleri ya da ender büyük oku/yaz hareketleri için ayarlanır. Diğer taraftan NoSQL, yoğun oku/yaz işlemlerini kolaylıkla sağlamaktadır.

NoSQL çözüm sağlayıcılarının genel olarak üzerinde birleştiği görüş; NoSQL'in mükemmel değil, cazip olduğu yönündedir. Birçok görüş NoSQL terimindeki No kelimesinin "Not Only" olduğu yönündedir. Yani sadece SQL değil. NoSQL'in SQL'i reddetmediği, yanı sıra ilişkisel veritabanlarının teknik limitlerini de dengelediği kabul edilir[6].

Dünya'da NoSQL örnekleri ele alındığında; sosyal ağlarda başkalarınca değerlendirilen unsurları gösteren belirleyiciler için Digg'in 3 TB'lı çözümü, Facebook'un gelen postaları arama için 50 TB ve eBay'in bütün verileri için 2 PB'lık çözümleri vardır. Yaklaşık on yıl önce, web uygulamalarının popüler hale gelmesi ile verilerin ölçeği üzerine çalışmalar yoğunlaşmıştır. Veritabanlarına ilişkin problemlerden biri olan ölçek sorununa, diğer çözümlerin içinde en iyi cevap vereni NoSQL'dir [4]. Günlük 7 TB'lık işlem hacmine sahip Twitter [5] ve 10 TB'lık Facebook örneğindeki gibi, çok büyük verilerin depolanması ve yazılmasında ilişkisel veritabanlarının problemleri için, birçok düğüm üzerine dağıtılarak yatay ölçekleme yapan dağıtık NoSQL çözümleri geliştirilmiştir.

ACID; IBM, DB2, MySQL, Microsoft SQL Server, PostgreSQL, Oracle uygulayıcıların kullandığı ya da çoğunun bilgi sahibi olduğu klasik ilişkisel veritabanı sistemlerinde sağlanan temel özelliktir ve Atomicity-Consistency-Isolation-Durability sözcüklerinin kısaltmasıdır [7].

ACID yapısı aşağıda verilen özellikleri kapsar:

- Bölünmezlik (**A**tomicity)
- Tutarlılık (**C**onsistency)
- İzolasyon (**I**solation)
- Dayanıklılık (**D**urability)

NoSQL mimarileri çoğu zaman sonunda tutarlı veya tek veri maddesiyle sınırlı işlemlerde zayıf tutarlılık garantisi verir. Fakat kimi sistemler, yardımcı özel yazılım tabakası ekleyerek bazı oluşumlarda tam ACID garantisi verirler (mesela CloudTPS).

İlişkisel veritabanlarının kullandığı ACID işlemselliğine karşın NoSQL BASE (Basically Available, Soft state, Eventually consistent) akronimi kısaltması ile ifade edilir [8].

- Kolay Ulaşılabilirlik (Basically Available): Veri erişim sorunlarını ortadan kaldırmak için kopyaları kullanır ve arta kalan herhangi bir paylaşılmış ya da bölümlenmiş veriyi birçok sunucudan alır.
- Esnek Durum (Soft state): ACID mantığında veri tutarlılığının olmazsa olmaz bir gereklilik olduğu savunulurdu fakat NoSQL sistemler tutarsız ve süreksiz verilerin barınmasına da izin verir.
- Eninde sonunda Tutarlı (Eventually consistent): Uygulamalar anlık tutarlılıkla ilgili olmasına rağmen, NoSQL sistemlerin gelecekte bir zamanda tutarlı olacağı farz edilir. ACID'in zorunlu tuttuğu tutarlılık, NoSQL'de tanımlanmayan bir zamanda tutarlılığın oluşacağı garanti edilir.

Sütun depoları için hızlı yalıtım sağlayan iki sistem geliştirilmiştir. Bunlar Google'un BigTable'a dayanan Percolator sistemi ve Waterloo Üniversitesi'nde HBase için geliştirilmiş hareketsel sistemi. Ayrı ayrı geliştirilmiş bu sistemler, Birbirine benzeyen hızlı yalıtım özellikleri kullanarak veri yönetimi için fazladan

işlemlere, ara yazılım yerleştirmesi veya ara yazılım tabakasından kaynaklanan bakıma gerek duymadan altındaki sütun deposu için garantili çok satırlı dağıtık ACID işlemleri sağlar. Dokümana dayalı depolama kategorisindeki NoSQL veritabanları, dokümanlara dönüştürülen anahtar değerlerden oluşan setler olarak tanımlanabilir. Her doküman benzersiz tekil bir anahtarla kimliklenir ve bir grupta toplanır. Bu dokümanlar XML veya JSON doküman türleri şeklinde olabilir. Veri erişimi belirli bir değer ya da anahtar üzerinden yapılabilir. Grafik (Graph) depolama kategorisindeki veritabanları ise veriler sosyal ağlardaki gibi grafik şeklinde temsil edildiğinde kullanılır [9].

Literatürde birçok NoSQL sınıflandırması bulunmaktadır. Ward' göre NoSQL, geniş veri setlerinde yüksek performanslı işlemleri optimize eden, depolama yazılımlarının alt kümesi için kullanılan popüler isim olup, yazılımları geliştiriciler için kullanımı kolay, yatay ölçeklemeye izin veren ve dar iş yükleri için optimizasyonu sağlayan bir veritabanı sistemidir. Ward, NoSQL' i üç ana kategoride toplamıştır [10]:

- Anahtar-Değer depolama
- Grafik depolama
- Doküman depolama

Genel olarak Ward'ın saydığı üç ana NoSQL kategorisine ek bir de sütuna dayalı depolama olarak karşımıza dördüncü bir kategori çıkmaktadır. Sütuna dayalı depolama kategorisinde aynen ilişkisel veritabanlarındaki gibi tablolar bulunmaktadır fakat bu tablolar çok daha esnektir. Sütunlar önceden tanımlı değildir ve satırlar bazen aralıklıdır [11].

NoSQL sistemleri genel olarak altı anahtar özelliğe sahiptir [12]:

- Yatay ölçekleme ile işleri birçok sunucuya yerleştirme yeteneği vardır.
- Verileri birçok sunucuya yedekleme ve dağıtma yeteneği vardır.
- Arayüzleri veya protokolleri kolayca çağırır (SQL'deki körlüğe karşılık).
- Birçok SQL veritabanın kullandığı işlemler arası katı ilişkiler daha zayıftır.
- Dağıtık indekslemeden dolayı verimli kullanım, veri depolama için verimli RAM kullanımı sağlar.

Veri kayıtlarına dinamik olarak farklı özellikler ekleyebilir. NoSQL veritabanları göreceli olarak yeni bir gelişmedir ve günümüzde ana akım kurumsal uygulamalar içinde kullanımı az bulunur. Fakat e-ticaret, internet arama motorları ve sosyal ağlar gibi büyük ölçekli internet uygulamaları için güvenilirliğini kanıtlamıştır. Birçok kayıt saklama teknolojisi kendilerini NoSQL ürünü olarak sınıflandırmaktadır ve her biri kendilerine özgü karakteristiklere, güce ve zayıflığa sahiptir [13].

Bilinen lider NoSQL ürünlerinin teknik karşılaştırmaları Tablo 1'de verilmiştir [6];

Tablo 1: Lider NoSQL ürünlerine ilişkin teknik karşılaştırma
(The Technical comparison about the leader NoSQL's products)

	MongoDB	**CouchDB**	**Riak**	**Redis**	**Voldemort**	**Cassandra**	**HBase**
Dil	C++	Erlang	Erlang	C++	Java	Java	Java
Lisans	AGPL	Apache	Apache	BSD	Apache	Apache	Apache
Model	Document	Document	Key/Value	Key/Value	Key/Value	Wide Column	Wide Column
Protokol	BSON	HTTP/REST	HTTP/REST or TCP/Proto	TCP		TCP/Thrift	HTTP/REST or
Depolama	Memory mapped b-trees	COW-BTree	Pluggable: InnoDB, LevelDB, Bitcask	In memory, snapshot to disk	Pluggable: BSV, MySQL, in-	Memtable/SStable	HDFS
Esinlenilen		Dynamo	Dynamo		Dynamo	BigTable, Dynamo	BigTable
Arama	Evet	Hayır	Evet	Hayır	Hayır	Evet	Evet
Sadeleştirme	Evet	Hayır	Evet	Hayır	Hayır	Evet	Evet

3. NOSQL KULLANMA NEDENLERİ (THE REASONS OF USE OF NOSQL)

NOSQL sistemlerin ne zaman, hangi koşullarda seçilmesi gerektiği hakkında standart kurallar olmamakla birlikte, farklı kaynaklardan edinilen bilgilerin derlemesi olacak biçimde aşağıdaki gibi sıralamak mümkündür.

- Değişken veri modeli ya da yapısı varsa,

- Veri bütünlüğü önemli değil ya da uygulama seviyesinde çözülmüşse,
- Mevcut ilişkisel modelin özellikle tablo birleştirme sorgularında getirdiği karmaşıklıktan kaçınmak üzere denormalizasyon işleminin yapılması gerektiği düşünülüyorsa,
- Mevcut sistem çoklu-nesne-hareketi (multi-object transaction) gerektirmeyen, daha çok her harekette bir nesne üzerinden işlem yapılıyorsa,
- Sistemin gereksinimleri sürekli değişkenlik gösteriyor ve bu değişiklikleri ilişkisel modele yansıtmak kayda değer kaynak maliyeti ortaya çıkarıyorsa,
- Mevcut sistemde aynı kayıtlar üzerinden okuma işlemi sık olarak gerçekleştiriliyor ve önbellek mekanizmasına ihtiyaç duyuluyorsa,
- Sistemde bulunan ve yapısal olmayan verilere ihtiyaç duyan, veri ambarı veya diğer uygulamalara sunulan verilerin saklanması için şema ve yapısal veri olgusu olmayan bir veritabanı ihtiyacı varsa Yüksek erişilebilirlik (high availability) adına yapılan harcamaların getirdiği performansa göre gereğinden fazla olduğu düşünülüyorsa, NOSQL sistemlerinin kullanılması düşünülebilir.
- Bulut bilişim servisleri ile organik bağlantıları vardır. Kolaylıkla bulut bilişimle birlikte kullanılabilir. Amazon gibi firmaların bu alanda çalışmaları bulunmaktadır. (Örnek http://aws.amazon.com/)

Packt Publishing yayınlarından Gaurav Vaish'in "Getting Started with NoSQL" kitabına göre dört temel özelliğini vurgulayarak neden NoSQL kullanmamız gerektiğini açıklamıştır [4]:

- Şablonsuz veri gösterimi (Schemaless data representation)**:** Hemen hemen tüm NoSQL uygulamaları şablonsuz veri temsilini sunar. Böyle bir yapıyı belirlemek için önceden düşünmek zorunluluğu yoktur ve zaman içinde geliştirmeye devam edilebilir. JSON kullanım avantajıyla, örneğin, veri yeni alanlar ekleme ya da iç içe geçme de dâhil olmak üzere birçok avantaj sağlar.
- Geliştirme zamanı (Development time): Karmaşık SQL sorguları ile uğraşmak zorunlu değildir. Bu da geliştirme zamanından kazanım sağlar. Veri tabanı üzerinde bir verinin görünümünü oluşturmak için birden çok tablo arasında verileri birleştirmek bağlamak zorunda kalınmamaktadır.
- Hız (Speed): Eldeki verilerin küçük bir miktarı ile, cep telefonu ve diğer aralıklı bağlanan aygıtlar üzerinden milisaniye hızında ve çok daha etkin veri dağıtımı sağlanabilir.
- Ölçeklenebilirlik için planlamanın önceden yapılması (Plan ahead for scalability**):** Bu kavramı doğru algılamak gerekmektedir. Geliştirilen uygulamalar oldukça esnektir. Ani yüklenmeleri çözebilir. Sadece tek bir planı gerçekleştirmek yerine esnek yapısından dolayı seçenek ve değişim imkânı sağlar.

NoSQL veri tabanları yaz/unut (fire&forget) prensibi ile çalışır. Bankacılık uygulamaları vb. için hiç uygun olmama sebebi de tam olarak budur. Buna göre veri, veritabanına gönderilir ve yazılıp yazılmadığı hakkında bilgi beklemeye gerek yoktur. Sosyal ağlarda oyun oynayanlar bu konuyu daha net tecrübe etmektedirler. Oyunlarında sürekli bir şeyler değiştirirler ve bunlar bir şekilde veri tabanına kayıt olur. Fakat bankacılık gibi sektörlerde bu özellik bir işe yaramaz ve hatta çok risklidir. Bunun için neredeyse tüm finans firmaları ACID kurallarını uygulamaktadırlar.

Beş ayrı sunucuda çalışıldığı varsayılırsa, "Birinci sunucuda güncellenen, acaba beşincide ne zaman güncellenecek?" sorusu akla gelebilir. Bu sorunun cevabı NoSQL geçici bellek (Ram) üzerinde çalıştığı için bilinemez. Yukarıda da değinildiği üzere, bankacılık uygulamaları gibi verinin kritik olduğu konularda kullanılamamaktadır.

NoSQL, "Number, String, Boolean, Array, Object, Null" veri tiplerini desteklemektedir.

MySQL, Cassandra, ve Hbase'in 7 özelliğe göre kıyaslanması Tablo 2'de şematize edilmektedir [14].

Tablo 2: Cassandra, HBase, Mysql Ürünlerinin 7 Özellik Çerçevesinde Gösterilmesi (7 characterized in that frame shown in of C,HB,M products**)**

Kıyaslamalar	**Cassandra**	**Hbase**	**MYSQL**
Kalıcılık (Persistance)	Evet	Evet	Evet (Farklı bir bağlantı yapısına sahip)
Çoğaltma (Replication)	Evet	Evet	Evet
Yüksek erişilebilirlik (High availaibity)	Dağıtık	Dağıtık	"Cluster" yapısı ile dağıtık
İşlemler (Transactions)	Olay bazlı tutarlılık	Yerel tutarlılık	Tutarlı (ACID)
Raf Tipi-Yerellik farkındalık (Rack-locality awareness)	Evet (Hadoop'tan kalıntılanmıştır)	Evet (Hadoop'tan kalıntılanmıştır)	Evet (MYSQL Cluster yapısı ile)
Uygulama Dili (Implementation Language)	Java	Java	ANSI-CI/ ANSI++

Etkililikleri/Sponsorlar (Influences/Sponsors)	Dynomo, BigTable Facebook/Digg Rackspace	BigTable	Oracle
Lisans Tipi (Licence Type)	Apache 2.0	Apache 2.0	GPL+FLOSS/Propriatery

4. KARŞILAŞTIRMALI ANALİZ (COMPARATIVE ANALYSIS)

NoSQL konusunda Türkçe literatüre katkı sağlamak amacıyla NoSQL ve ilişkisel veritabanlarını karşılaştırılmıştır. İki sistemimin kullanımı konusunda karar vermek için mini bir analiz yapılmıştır. Analizde en yaygın kullanılan MongoDB ve MySQL sistemleri esneklik, veri bağlantıları, performans ve tutarlılık gibi 4 ana başlık altında ele alınmıştır.

4.1. Esneklik (Flexibility)

Esneklik, NoSQL veritabanlarının en büyük artılarından birisi sayılabilir. İlişkisel veritabanlarının sınırları sabittir. Belli sütunlar üzerinde ekleme çıkarma yapılabilir. Ama NoSQL veritabanlarında istenilen sütün eklenebilir, istenilen sütuna ise yazılmayabilir. Bu özellik, yazılımcı ve tasarımcı açısından genellikle bir avantaj gibi görünse dahi kimi zaman zorunlu olarak doldurulması gereken bir sütun gözden kaçabilir ve istenilen bilgilere ulaşılamayabilir. Aşağıdaki örnekte olduğu gibi C# ve linq dilinde yazılım geliştirmeye yatkın olanlar mongo-db sürücü ile kolaylıkla dil üzerinde dökümanlarını saklayabilir ve arayabilirler.

Örnek:

```
var query =
        from e in collection. AsQueryable<ISCI>()
        where e.FirstName  == "Ali" && e.LastName == "Yılmaz"
        select e;
foreach (var  isci in query)
{
        // Adı Ali Soyadı Yılmaz olan işçilerin listesi üzerinde yapılacak işlemler
}
```

4.2 Veri Bağlantıları - İşkisel Yönetim (Data Links – Relational Management)

Burada İVTYS özelliği öne çıkarılabilir. NoSQL tarafında bu özellik yoktur ve uygulama seviyesinde çözümlenmelidir.

4.3. Performans (Performance)

İki veritabanı arasındaki diğer bir fark oluşturacak unsur performanstır. Artan yazılım rekabetinde kullanıcının yazılımı tercih etmesinin ilk sebeplerinden birisi çalışma hızıdır. Bu neden göz önünde bulundurularak en yaygın olarak kullanılan MongoDB ve MySQL sistemlerinin performanslarının karşılaştırılması için olabilecek en eşit şartlarda aynı bilgisayar üzerinde kayıt ekleme ve sorgulama işlemlerinin süreleri hesaplanmıştır.

Öncelikle sistemin testlerinin doğru bir şekilde yapabilmesi için MySQL ve MongoDB de çeşitli ayarlamalar yapılmıştır.

Örnek:

10000000 Kayıt Ekleme

Kayıt ekleme sırasında farklı veri tiplerinde(string,

Veritabanı	Süre
MongoDB	**960090** milisaniye
MySQL	**2640037** milisaniye

10000000 kayıt içinden 500 kayıt okuma (İlk Çalıştırma)

Veritabanı	Thread	Süre
MongoDB	1	**18200** millisaniye
MySQL	1	**6276** millisaniye

10000000 kayıt içinden 500 kayıt okuma

Veritabanı	Thread	Süre
MongoDB	1	**5750** millisaniye
MySQL	1	**480** millisaniye
MongoDB	2	**11002** millisaniye
MySQL	2	**504** millisaniye
MongoDB	4	**62021** millisaniye
MySQL	4	**634** millisaniye

10000 kayıt güncelleme

Veritabanı	Veri Türü	Süre
MongoDB	Metin	**86396** millisaniye
MySQL	Metin	**10252** millisaniye
MongoDB	Ondalıklı Sayı	**106668** millisaniye
MySQL	Ondalıklı Sayı	**10486** millisaniye
MongoDB	Doğru/Yanlış	**107992** millisaniye
MySQL	Doğru/Yanlış	**10885** millisaniye

Performans sonuçlarına bakıldığında MongoDb'nin kayıt ekleme hızının MySQL'e göre 3 kata yakın daha hızlı olduğu söylenebilir. Kayıt okuma hızlarına bakıldığında ise sorgunun ilk çalıştırılmasında MySQL'in MongoDB'ye göre 3 kat daha hızlı olduğu söylenebilir. Sonraki çalıştırmalarda bu oran 10 katın üstüne çıkmaktadır. Eğer aynı işlem birden fazla kanal kullanarak çoğaltılır ve denenirse bu fark 4 kanallı bir sistemde 100 katına kadar çıkmaktadır.

Güncelleme de MySQL in MongoDB ye göre farklı veri türlerinde daha hızlı işlem gerçekleştiğini görüyoruz.

Performans değerlendirmesi sırasında yazılımın veritabanına bağlanma şekli, yazılım üzerinde yapılacak indekslemeler veya sorgu sırasındaki çekilecek anlık veri miktarına göre bu performans süreleri farklılık gösterebilir.

4.4. Tutarlılık (Consistency)

Veritabanları üzerinde yapılacak diğer bir karşılaştırma unsuru tutarlılıktır. NoSQL sistemi üzerinde bilinen türde

bir ilişki yapısı olmaması nedeniyle art arda bağlantıları olan işler yapıldığı zaman bütünsel işlem kavramı olmadığından oluşan bir hatadan dönmek son derece zor olabilir. Ancak ilişkisel bir veritabanında yapılacak basit kontrollerle bu tutarlılık sağlanabilir. Özellikle finans sektöründe ve parasal verilerin olduğu veritabanlarında tablolar arası uyum son derece önemlidir. Bu nedenle bu tür veriler için ilişkisel veritabanlarını kullanmak daha doğru olabilir. Diğer türlü yazılım katmanında ağır bir iş yüküne ve hata kontrol sistemine ihtiyaç vardır.

4.5. Sorgulama Dili (Query Language)

NoSQL veritabanları ile SQL veritabanları kullanılan dil bakımından da farklılık göstermektedir. SQL kullanan ilişkisel veritabanlarının çoğu aynı dil yapısına sahiptir. Ancak NoSQL veritabanlarının kendine ait farklı sorgulama dilleri bulunabilmektedir. MongoDB ve MySQL veritabanlarını dil bakımdan karşılaştırmak için Musteriler(ad,soyad,telefon) tablosunu kullanacağız.

Tablo 3: Dil bakımından İlişkisel Veritabanları ile MongoDB eşleştirmesi

SQL	**MongoDB**
database	database
table	collection
row	document veya BSON document
column	field
index	index
primary key	primary key

4.5.1. Ekleme (Insert)

MySQL→ insert into Musteriler(ad,soyad,telefon) values (‘Ali,’Yılmaz’,’452256235’)
MongoDB→ db.Musteriler.insert({ad:”Ali”, soyad:”Yılmaz”, telefon:” 452256235”})

4.5.2. Seçme (Select)

MySQL → select * from Musteriler
MongoDB → db.Musteriler.find()

MySQL → select * from Musteriler where Ad=’Ali’
MongoDB → db.Musteriler.find(ad:”Ali”)

4.5.3. Silme (Delete)

MySQL → delete from Musteriler
MongoDB → db.Musteriler.remove()

MySQL → delete from Musteriler where Ad=’Ali’
MongoDB → db.Musteriler.remove(Ad:”Ali”)

4.5.4. Güncelleme (Update)

MySQL → update set telefon=‘43534634’ Musteriler where ad=’Ali’
MongoDB → db.Musteriler.update(
{ ad: "Ali" },
{$set:{ telefon: ‘43534634’} }, { upsert: true })

5. SONUÇ VE DEĞERLENDİRME (RESULT AND EVALUATION)

Yüksek internet hızı ve çok sayıda kullanıcının zorladığı teknolojik sınırlar insanları değişik çözümler üretmeye itmektedir. Günümüzde değişken ve büyük veri yapıları için NoSQL kullanılabilecek yegâne teknolojidir. Oracle'ın da NoSQL dünyasında olduğu düşünüldüğünde konunun önemi daha rahat anlaşılacaktır.
NoSQL veritabanları büyük miktarda veri yönetmek için daha az yapılandırılmış bir yol sunar. Bunun bir örneği anahtar-değer veritabanıdır. Temel veriler depolanır, aranabilir, ve değer verilerin geri kalanının nerede bulunabileceğine dair bir referans olur. İlişkisel veritabanlarında verilerin yapılandırılabilmesi için bazı kısıtlamalar vardır, ancak NoSQL de bu kısıtlamaların daha az olması nedeniyle NoSQL veritabanları daha esnektir diyebiliriz. İlişkisel veritabanları bu tür büyük miktarda veri depolamak için, olumsuz farklı parçalar arasındaki bağlantıları sorgulama ve bulma kolaylığı gibi birçok avantajlara sahip olsa da tam olarak oturmamış bir sistemdir. Örneğin Ad hoc sorguları zordur, yeni ve sürekliliği devam eden teknolojiler olduğu için bu konuda uzmanlaşma hala sürmektedir. Fakat NoSQL veri tabanları birçok farklı türde büyük miktarda veri yönetimi için esnek yollar sunar. İnternet ortamında hızla artan veri yoğunluğu düşünüldüğünde geleceğin teknolojisi ilişkisel veri tabanları yerine NoSQL veri tabanı teknolojisi olabileceği düşünülmektedir.
Çeşitli açılardan iki veritabanı yaklaşımı göz önüne alındığında ikisinin de fark yarattığı, ve ön plana çıktığı noktalar olduğu görülmektedir. Ancak bazı açılardan ilişkisel bir veritabanının kullanılmasının çok daha doğru olduğu görülmektedir. Bu açıdan ele alınırsa sektörel bazlı düşünmek ve farklı sektörlerde işlemlere bağlı olarak farklı veritabanı sistemlerinin önerilmesi doğru olacaktır. Kimi uygulamalarda bu iki sistemin birlikte çalışması da sağlanabilir ve hali hazırda piyasada bu şekilde çalışan sistemlerin olduğu unutulmamalıdır. Sabit olarak tutulan, yüksek oranda okuma ve yazma yapılan tablolar (örneğin müşteri tablosu) NoSQL veritabanlarında tutulurken, daha çok finans bilgilerinin tutulduğu hesaplama içeren, birkaç tablo üzerinde işlem yapan bilgiler (örneğin müşteri hareketleri tablosu) ilişkisel veritabanlarında tutulabilir. Bu şekilde bu iki sistemin de ön plana çıkan özellikleri kullanılabilir.

KAYNAKLAR (REFERENCES)

[1] V. Mayer-Schönberger, K. Cukier , "**Big Data: A Revolution That Will Transform How We Live, Work, and Think**", Houghton Mifflin Hartcourt, USA, 2013

[2] J. CELKO, "**Joe Celko's Complete Guide to NoSQL: What Every SQL Professional Needs to Know about Non-Relational Databases**", Newnes, 2013

[3] C. Mohan, "**History Repeats Itself:Sensible and NonsenSQL Aspects of the NoSQL Hoopla**", EDBT/ICDT '13, March 18–22, Genoa, Italy, 2013

[4] G. VAISH, "**Getting Started With NoSQL**", Packt Publishing, United Kingdom, 2013

[5] M. Otey, "**NoSQL? No Way!" ,**SQL Server Magazine , pp:5, 2010

[6] G. BURD, "**NoSQL**", Sysadmin Journal, Vol:36 No. 5 ss:5-12 ABD, 2011

[7] Internet: http://en.wikipedia.org/wiki/Acid; 08.08.2014

[8] An Oracle White Paper, "**Oracle NoSQL Database**" , 2011

[9] V. Abramova, J. Bernardino, "NoS**QL Databases: MongoDB vs Cassandra**" ,Conference C3S2E, Porto, Portugal pp:14-22, 2013

[10] M. Ward, "**NoSQL Database in the Cloud: MongoDB on AWS**", Amazon Web Services E-Journal, 2-22 March 2013

[11] P. Atzeni, F. Bugiotti, L. Rossi, "**Uniform access to NoSQL systems**" Elsevier Ltd., http://dx.doi.org/10.1016/j.is.2013.05.002, 2013

[12] R. Cattell, "**Scalable SQL and NoSQL Data Stores**" ACM SIGMOD Record Volume 39 Issue 4, Pages 12-27 USA, 2010

[13] N. ROZANSKI, E. WOODS, "**Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives**", 2nd Edition Pearson EDucation Inc., USA, 2012

[14] B. G. Tudorica, C. Bucur, "**A comparison between several NoSQL databases with comments and notes**", Department for Economical Mathematics and Economical Informatics, Petroleum-Gas University of Ploiesti Ploiesti, Romania, 2011.