Mobil Cihazlarda Gömülü Veritabanlarının Karşılaştırılması: SqLite ve CouchBase Lite

Erkan Güler[1], Taner Arabacıoğlu[2], Özel Sebetci[3]

[1] Adnan Menderes Üniversitesi, Bilgisayar Teknolojileri ve Programlama Programı, Aydın

[2] Adnan Menderes Üniversitesi, BÖTE, Aydın

[3] Adnan Menderes Üniversitesi, Bilgisayar Teknolojileri ve Programlama Programı, Aydın

eguler@adu.edu.tr, tarabacioglu@adu.edu.tr, osebetci@adu.edu.tr

Özet: Bilişim teknolojilerindeki hızlı değişim sonucu veri kullanımı ve depolanması gereken veri miktarı artmaktadır. Sosyal ağlar ve bloglar gibi Web 2.0 uygulamaları internette dolaşan yapılandırılmamış verilerin artışındaki temel öğeler olarak söylenebilir. 2015 yılında mobil veri trafiğinin ulaşması beklenen büyüklüğü 75 exabyte olması ve bu rakamın 2000 yılındaki toplam internet trafiğinin 75 katı olması depolanacak verinin büyüklüğü ve artış miktarı konusunda fikir vermektedir. Yaklaşık 40 yıl öncesinde tasarlanan ilişkisel Veri Modelinin günümüzde büyük veriler için yetersiz kaldığı, NoSql veritabanlarının ortaya çıkışıyla doğrulanmaktadır. Bu çalışmada, doküman tabanlı veritabanlarından olan CouchBase Lite ve ilişkisel veritabanı SqLite'ın mobil cihazlardaki gömülü olarak kullanım karşılaştırması amaçlanmıştır. Android Studio ile geliştirilen uygulama ile 5000 kayıt söz konusu veritabanlarına kayıt edilmiş ve okunmuştur. Elde edilen sonuçlara göre doküman oluşturma işlemi, tablo oluşturmaya oranla yaklaşık iki kat daha hızlı gerçekleşirken, veri okuma hızında ilişkisel veritabanı az bir fark ile olsa da daha iyi bir performans göstermiştir. Mevcut sonuçlar, veritabanı güvenliği ve semantik web açısından tartışılmıştır.

Anahtar Sözcükler: Veritabanı Yönetim Sistemleri, Sqlite, CouchBase Lite

1. Giriş

Geleneksel dosyalama sistemlerinden bu yana verilerin saklanması her dönemde önemli bir yer tutmuştur. Ancak günümüzde sosyal ağlar ve bloglar gibi Web 2.0 uygulamalarının milyonlarla ifade edilen kullanıcılara ulaşması internette dolaşan veri miktarının beklenmedik şekilde artmasına neden olmuştur. Cisco'nun yayınladığı Görsel Ağ Endeksi Öngörü Raporu'na (2015) göre önümüzdeki 5 yıl içinde mobil veri trafiğinin 10 katına çıkacağı, 2019 yılında akıllı cihazlar ve bağlantılarının küresel mobil trafiğin yüzde 97'sini oluşturacağı düşünülmektedir. Veri trafiğinin boyutları noktasında ise 2015 yılında mobil veri trafiğinin ulaşması beklenen büyüklüğün 75 exabyte olması

ve bu rakamın 2000 yılındaki toplam internet trafiğinin 75 katı olması dikkati çeken diğer bir husustur. Bu bağlamda depolanacak verinin artış miktarı ve mobil cihazların söz konusu verideki kullanımı gelecek adına önemli fikirler vermektedir.

Böylesine bir gelişim ve kullanım süreci günümüzde sık kullanılan ilişkisel veri modelini de değişime zorlamaktadır. Özseven'e (2011) göre temeli 1970'lere dayanan ve 1985'ten sonra yaygınlaşan modelin günümüz beklentilerini karşılamada yeterli olamayacağı eldeki veriler doğrultusunda söylenebilir. Bu noktada ilişkisel veri modelinin dışında NoSql olarak adlandırılan veritabanlarının ortaya çıkışı ve kullanımı görülmektedir. Oracle (2012) NoSql veritabanlarının ortaya çıkışı için çok büyük verilerle uğraşan ve gecikme sürelerine ilişkin sıkı kısıtlamaları bulunan Amazon, Google, Linkedin ve Twitter'ı işaret etmektedir.

Ünalır ve diğerlerine (2015) göre, NoSql veri modelleri anahtar-değer, çizge, doküman ve sütun veritabanı olmak üzere 4 alt kategoriye ayrılır. Anahtar-değer veritabanı okuma ve yazma işlemlerini hızlandırır. Çizge veritabanı düğümlerin bağlar aracılığıyla kolay bir şekilde dolaşılmasını sağlar. Doküman veritabanı farklı biçimlerdeki veri kaynaklarının daha esnek işletilmesine olanak tanır. Sütun veritabanı ilişkili sütunların bir sütun ailesinde toplanmasıyla uygulamalara geniş kapsamlı sorgu ve veri çözümlemeleri yapabilme yeteneğini sunar.

Mohammed, Altrafi ve Ismail (2014) ilişkisel veritabanlarını 9 faklı özelliğe göre NoSql veritabanları ile karşılaştırdığında aşağıdaki sonuçlara ulaşmışlardır:

İşlem güvenilirliği (Transaction reliability): ACID kuralları ile çalışan ilişkisel veritabanları bu noktada NoSql veritabanlarına üstünlük sağlamaktadır.

Veri Modeli (Data Model): İlişkisel veritabanları işlemler matematiksel olarak modellenir. Sütunlar iyi tanımlanmıştır ve ilişkili veriler aynı yapıdaki satırlarda saklanır. Bu iyi organize edilmiş bir veri modelidir. NoSql ise kategorilere ayrılmasını sağlayan teknikleri veri modeli olarak kullanır. En belirgin ayrım ise, depolama yapısı olarak tablolar kullanmaz. Bu durum Word, pdf, resim ve video gibi yapılandırılmamış verinin yönetilmesinde etkilidir.

Ölçeklenebilirlik (Scalability): İlişkisel veritabanları dikey olarak ölçeklenebilirken, NoSql veritabanları yatay olarak ölçeklenebilir. Dikey büyümenin beraberinde getirdiği problem ise daha fazla donanım kaynağıdır.

Bulut (Cloud): İlişkisel veritabanları veri aramada tüm içeriği birden desteklemez ve verileri bir limitin ötesinde ölçeklendirmek zordur. NoSql veritabanları ise yapılandırılmamış, yarı yapılandırılmış ya da yapılandırılmış veriler için esnek bir yapıdadır.

Büyük veriyi işleme (Big Data Handling): İlişkisel veritabanlarında veriler dikey olarak büyüdüğü için yeni bu durum yeni sunucular ve performans sorunları ortaya çıkarabilmektedir. NoSql ise büyük veriler için tasarlandığı için performans problemleri yaşamamaktadır.

Veri ambarı (Data warehouse): İlişkisel veritabanlarında depolanan veri zamanla artmakta ve OLAP, veri madenciliği ve istatistiksel işlemler yoluyla bu problem aşılmaya çalışılmaktadır. NoSql ise veri ambarı olmak için tasarlanmamış ve tasarımcılar yüksek performans ve ölçeklenebilirliğe odaklanmışlardır.

Karmaşıklık (Complexity): İlişkisel veritabanında kullanıcılar verileri tablolara dönüştürmek zorundadır. Veri-tablo uyumsuzluğu yaşandığında, veritabanının yapısı karmaşık hale gelebilir. NoSql veritabanlarının ise yapılandırılmamış, yarı yapılandırılmış ya da yapılandırılmış verileri saklama yeteneğinden dolayı böyle bir sorun yaşanmaz.

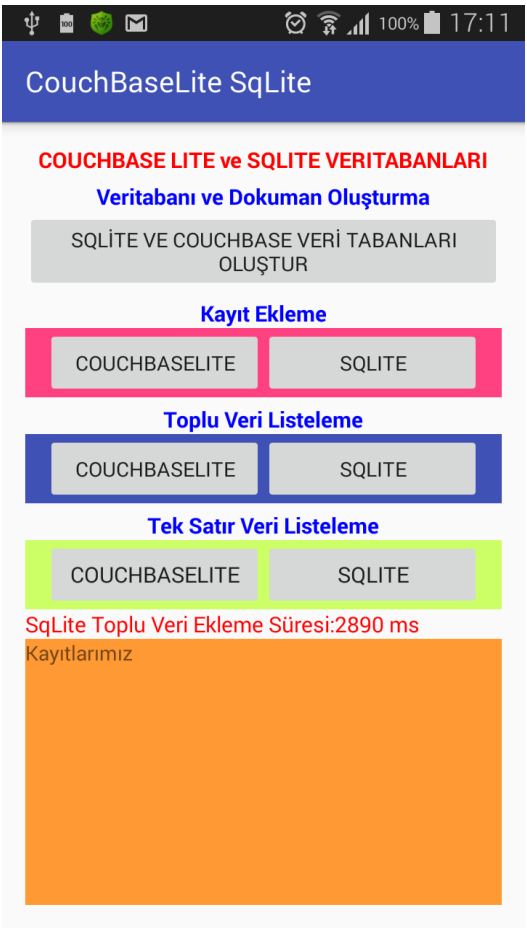
Çökme kurtarıcı (Crash recovery): İlişkisel veritabanları log dosyaları ve ARIES algoritmaları sayesinde yaşanabilecek veri kayıplarını önler. Ancak NoSql veritabanları için veri kurtarma yedek alma sayısına bağlı değişir.

Güvenlik (Security): İlişkisel veritabanları birçok güvenlik servisini desteklemekte ve bu konuda birçok çalışma bulunmaktadır. Birçok NoSql veritabanı ise güvenlik sorununu çözmeye çalışmaktadır.

Bu çalışmada, mobil cihazlarda gömülü olarak kullanılan SqLite ilişkisel veritabanı ile CouchBase Lite NoSql veritabanının karşılaştırılması amaçlanmaktadır. Araştırmada kullanılan NoSql veritabanı olan CouchBase Lite, anahtar-değer depolamaya oldukça benzerlik göstermektedirler. Veri tutma modeli, anahtar-değer çiftlerinin toplanmasından oluşan versiyonlanmış haldeki dokümanlardır. Yarı yapılandırılmış dokümanlar JSON (JavaScript Object Notation) formatında tutulur (Eken, Kaya, Sayar & Kavak, 2014).

2. Yöntem

Söz konusu veritabanlarının karşılaştırılabilmesi amacıyla Windows 10 64 Bit işletim sistemi kurulu bir dizüstü bilgisayarda Android Studio 1.4 kullanılarak, Java programlama dilinde geliştirilmiştir. Geliştirilen uygulamanın arayüzü Şekil 1'de verilmiştir.



Şekil 1. Uygulama arayüzü

Uygulamada SQLite ve CouchBase Lite veritabanları kullanılarak veriler düzenlenmiştir. Android sürümü olarak en düşük Android 2.2 işletim sistemi ve API 8 seviyesi mobil cihazlarda çalışacak şekilde tasarlanmıştır. Veritabanı oluşturma, kayıt ekleme, toplu veri listeleme ve tek satır veri listeleme bölümlerinden oluşan uygulama, tek bir doküman ya da tablo üzerinden sorguladığı verilerin işlenme sürelerini *ms* cinsinden ekrana yansıtmaktadır. Uygulama geliştirmenin anahtar kodları her bir adım için aşağıda verilmiştir:

Veritabanı ve doküman oluşturmak için;

```
vt = new Veritabani(MainActivity.this);
document1 = database.createDocument();
```

Veri eklemek için;

```
long id = db.insert("kayitlar", null, insertValues);
document1.putProperties(props);
```

Veri listelemek için;

```
Cursor kayitlar = db.rawQuery("SELECT * FROM kayitlar", null);

kayitListesi.setText(String.valueOf(retrievedDocument.getProper
ties()));
```

Tek bir kayıt okumak için;

```
Cursor kayitlar = db.rawQuery("SELECT * FROM kayitlar where
key='"+_key+"'", null);
String owner = (String) properties.get("key4");
```

3. Bulgular

SqLite ve CouchBase Lite'ın mobil cihazda gömülü olarak kullanım performanslarını belirlemek amacıyla üstte belirtilen adımlar 5000, 10000 ve 20000 kayıt için tekrarlanmıştır. Her bir veri sayısı için aynı işlem 5 kez art arda yapılarak aritmetik ortalaması alınmıştır. İlk testte 5000'lik veri üzerinde çalıştırılan komutlar ve sonuçları Tablo 1 de verilmiştir. Tablo 2 ve Tablo 3'de de sırasıyla 10000 ve 20000 kayıt için performans verileri listelenmiştir.

Tablo 1: 5000 kayıt için performans verileri (ms)

Deneme Sayısı	DataBase Oluşturma		Kayıt Ekleme		Toplu Veri Listeleme		Tek Kayıt Listeleme	
	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite
1	111	38	314	855	3571	2778	21	19
2	104	44	305	846	4432	2752	22	21
3	105	51	301	870	3811	2756	22	20
4	102	45	299	903	4443	2756	23	19
5	114	35	305	902	3818	2928	22	19
Ortalama	**107,2**	**42,6**	**304,8**	**875,2**	**4015**	**2794**	**22**	**19,6**

Tablo 2: 10000 kayıt için performans verileri (ms)

Deneme Sayısı	DataBase Oluşturma		Kayıt Ekleme		Toplu Veri Listeleme		Tek Kayıt Listeleme	
	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite
1	109	59	604	1989	7417	5564	40	30
2	104	41	564	1574	7390	5628	41	32
3	102	38	531	1508	7459	5596	42	29
4	105	43	520	1513	7525	5572	41	28
5	111	41	526	1496	7484	5532	41	26
Ortalama	**106,2**	**44,4**	**549**	**1616**	**7455**	**5578,4**	**41**	**29**

Tablo 3: 20000 kayıt için performans verileri (ms)

Deneme Sayısı	DataBase Oluşturma		Kayıt Ekleme		Toplu Veri Listeleme		Tek Kayıt Listeleme	
	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite	CouchBase Lite	Sqlite
1	118	49	1216	3175	15507	11746	66	46
2	122	40	958	2849	15486	11574	66	52
3	109	37	998	2823	15503	11414	64	47
4	110	39	958	2861	15441	11467	68	49
5	103	49	943	2808	15512	11701	62	46
Ortalama	**112,4**	**42,8**	**1014,6**	**2903,2**	**15489,8**	**11580,4**	**65,2**	**48**

Her iki veritabanında tek bir döküman ve tablo kullanılarak yapılan hız testlerinde kayıt ekleme dışında kalan veritabanı ya da doküman oluşturma, tüm kayıtları listeleme ve tek kaydın seçim işleminde SqLite'ın üstünlüğü göze çarpmaktadır. Her üç veri büyüklüğünde de benzer sonuçlar elde edilmesi, ilişkisel veritabanlarının hala kullanılabilirliğinin göstergesi olarak yorumlanabilir.

4. Sonuç

NoSql veritabanlarının ilişkin büyük verilerdeki performasının SqLite ilişkisel veritabanı ile karşılaştırıldığı çalışma sonucunda literatür sonucu oluşan beklentilerin gerçekleşmediği söylenebilir. Bu durum için sadece bir doküman

ve tablo ile testlerin gerçekleştirilmesi nedeniyle, bir sınırlılık olarak değerlendirilmektedir. Sonraki çalışmalar için bu hususun göz önünde bulundurulması, farklı NoSql veritabanlarının kullanılması ve online bir bağlantı ile testlerin tekrar edilmesi önerilmektedir.

KAYNAKÇA

Eken, S., Kaya, F., Sayar, A. & Kavak, A. (2014, Mayıs) *" Doküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB yatay ölçeklenebilirlik karşılaştırması"*, 7. Mühendislik ve Teknoloji Sempozyumu , Ankara-Türkiye.

Mohamed M, A., Altrafi O, G. & Ismail M, O. (2014). Relational vs. NoSQL Databases: A Survey. *International Journal of Computer and Information Technology,* 3 (3), 598-601.

Ünalır, T.O., İnan, E., Yöngül, B., Olca, E., Şentürk, F. & Mostafapour, V. (2015, Eylül) *"Veri Yoğun Bilgi Sistemleri için Melez Bir Veri Mimarisi Önerisi"*, 9. Ulusal Yazılım Mühendisliği Sempozyumu , 90, İzmir-Türkiye.

http://www.cisco.com/web/TR/news/press/archive/2015/120215.html Erişi tarihi:10.11.2015

http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-wp-1436762.pdf Erişi tarihi:10.11.2015