Araştırma Makalesi / Research Article

MongoDB'nin ilişkisel veri tabanları gibi kullanılabilmesi

Anıl YILDIZ[1]
Parvaneh SHAMS[2]
Zafer GÜNEY[3]

Geliş tarihi / *Received:* 21.09.2021
Düzeltilerek geliş tarihi / *Received in revised form:* 19.11.2021
Kabul tarihi / *Accepted:* 21.10.2021
DOI: 10.17932/IAU.ABMYOD.2006.005/abmyod_v16i63002

Öz

Son 20 yılda teknoloji çok büyük bir hızla gelişmiştir. Gelişen teknoloji ile birlikte dijital ortamdaki verilerin boyutlarında da çok ciddi bir artış yaşanmıştır. Geleneksel olan ilişkisel veri tabanları dijital ortamdaki veri miktarlarını verimli bir şekilde saklayabilecek ve işlem yapabilecek gelişmeyi göstermede yeterlilik sağlayamamıştır. İlişkisel veri tabanları veri tabanı içerisindeki bütün verileri birbiri ile ilişkilendirebilirler ve gelişmiş sorgulama dili aracılığıyla kullanıcılarına detaylı sorgular, raporlar üretebilirler. Ancak ilişkisel veri tabanları performans olarak ve mali olarak ilişkisel olmayan veri tabanlarının oldukça gerisinde kalmıştır. İlişkisel veri tabanlarının ekonomik açıdan çok da verimli olmaması ve artan dijital veri miktarına bağlı olarak veri tabanı performansında da ciddi düşüşler göstermesi ilişkisel olmayan veri tabanlarının kullanımını

[1] *Yüksek Lisans Öğrencisi, İstanbul Aydın Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Küçükçekmece/İst., e-mail: anily@aydin.edu.tr, ORCID ID: 0000-0003-4607-6660*
[2] *Dr. Öğr. Üyesi, İstanbul Aydın Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Küçükçekmece/İst.,e-mail: parvanehshams@aydin.edu.tr, ORCID ID: 0000-0003-1467-3284*
[3] *Dr. Öğr. Üyesi, İstanbul Aydın Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi, Küçükçekmece/İst., e-mail: zaferguney@aydin.edu.tr, ORCID ID: 0000-0003-1974-4264*

ciddi şekilde arttırmıştır. İlişkisel olmayan veri tabanları özel olarak hazırlanmış yapılar olup performansı, kolay sürdürülebilirliği ve uygun maliyetli olması ile popülarite kazanmıştır. İlişkisel olmayan veri tabanları her ne kadar yapısı gereği ilişkisel olarak kullanılamasa da günümüzün gelişmiş programlama dilleri aracılığıyla ve gelişmiş bir veri tabanı mimarisi ile birlikte ilişkisel olarak kullanımı mümkündür. Bu çalışmada da günümüzün en gelişmiş ve en popüler ilişkisel olmayan veri tabanlarından birisi olan MongoDB'nin C# programlama dili aracılığıyla nasıl ilişkisel olarak kullanılabileceği ve bunun bizlere ne gibi kazançlar sağlayacağı üzerinde durulmuştur.

Anahtar Kelimeler: *İlişkisel veri tabanı, ilişkisel olmayan veri tabanı, büyük veri, MongoDB, NoSQL*

Using MongoDB like relational databases

Abstract

Technology has developed rapidly in the last 20 years. Along with the developing technology, there has been a significant increase in the size of the data in the digital environment. Traditional relational databases have not been able to efficiently store and process the amount of data in the digital environment. Relational databases can associate all the data in the database with each other and can generate detailed queries and reports for their users through the advanced query language. However, relational databases lag far behind non-relational databases in performance and financially. The fact that relational databases are not economically efficient and that the database performance decreases due to the increasing amount of digital data has seriously increased the use of non-relational databases. Non-relational databases are specially prepared structures and have gained popularity with their performance, easy maintainability,

and cost-effectiveness. Although non-relational databases cannot be used relationally due to their structure, it is possible to use relationally with today's advanced programming languages and advanced database architecture. In this study, it is emphasized how MongoDB, one of the most advanced and most popular non-relational databases of today, can be used relationally through the C# programming language and what benefits this will provide for us.

Keywords: *Relational database, non-relational database, big data, MongoDB, NoSQL*

Giriş

Dijital veri boyutlarının çok yüksek seviyelere ulaşmasıyla birlikte ilişkisel veri tabanlarının da bu artışa paralel olarak ciddi performans kayıpları yaşandığı gözlemlenmiştir (Öztürk ve Atmaca, 2017; Patil ve ark., 2017). İlişkisel olmayan veri tabanlarının kullanımı her ne kadar ilişkisel veri tabanlarına göre daha zor olsa da performans ve maliyet yönünden çok daha etkin bir çözüm sunabilmektedir (Öztürk ve Atmaca, 2017; Yuan ve ark., 2018). İlişkisel olmayan veri tabanları veri üzerinde yapılan okuma ve yazma gibi işlemlerde ilişkisel veri tabanlarına göre çok daha başarılı oldukları için de eBay, Amazon ve Facebook gibi büyük şirketler tarafından da tercih edilmektedir (Boicea ve ark., 2012; Jung ve ark., 2015; Öztürk ve Atmaca, 2017; Yuan ve ark., 2018).

İlişkisel olmayan veri tabanları yapısı gereği ilişkisel yapıda kullanılmamaktadırlar. Ancak günümüzün gelişmiş programlama dilleri vasıtasıyla ilişkisel veri tabanlarının kullanıma benzer bir yapıda kullanılabilmektedir. İlişkisel olmayan veri tabanlarının ilişkisel yapıda kullanılabilmesiyle birlikte performans ve maliyet anlamında ciddi kazanımlar elde edilmesi mümkündür (Öztürk ve Atmaca, 2017; Yuan ve ark., 2018). Bu çalışma ilişkisel olmayan bir veri tabanının daha karmaşık

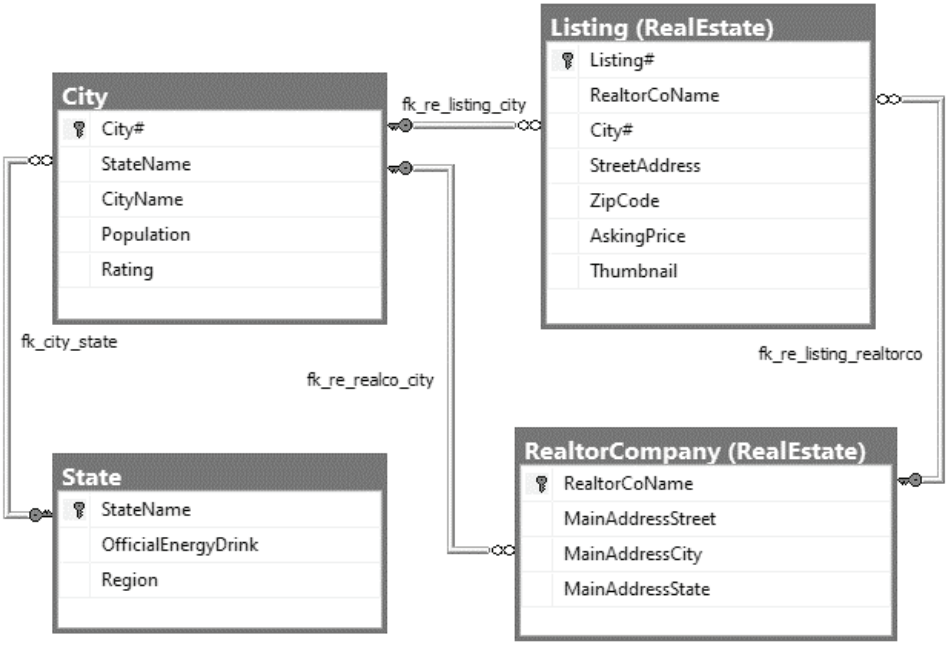
veri tabanı yapılarında dahi başarılı bir şekilde çalışabileceğini ve bunun ne gibi avantajlar sağlayacağını ifade etmeyi amaçlamıştır.

İlişkisel veri tabanı

Veri tabanı en genel tanımıyla içerisinde verilerin saklandığı ve bu veriler üzerinde çeşitli işlemlerin gerçekleştirilebildiği yapılardır. Veri tabanları temelde ilişkisel ve ilişkisel olmayan veri tabanları olarak ikiye ayrılırlar (Boicea ve ark., 2012; Öztürk ve Atmaca, 2017; Yuan ve ark., 2018). İlişkisel veri tabanlarının çalışma mantığı veriler arasında ilişki kurmaya dayanır. İlişkisel veri tabanları verileri tablolarda tutar. Şekil 1'de bazı ilişkisel veri tabanlarının görseli verilmiştir (URL 1).

Şekil 1. *Bazı ilişkisel veri tabanları (URL 1)*

İlişkisel veri tabanları, veri tabanları içerisindeki bütün verileri birbiri ile ilişkilendirme imkanına sahiptirler ve gelişmiş sorgulama dilleri ile kullanıcılarına detaylı sorgular, raporlar üretebilmektedirler (Öztürk ve Atmaca, 2017). Şekil 2'de örnek bir ilişki yapısı verilmiştir (URL 2).

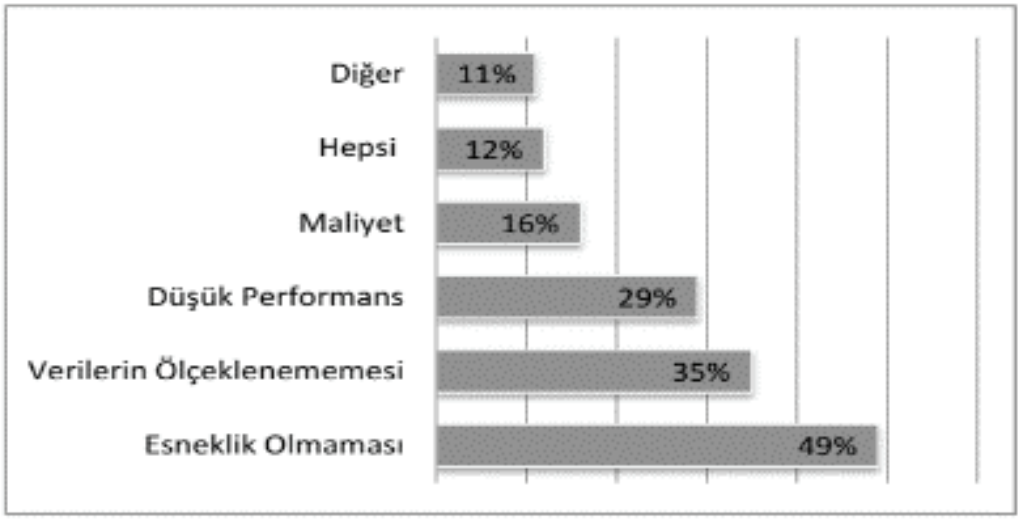


Şekil 2. *Örnek bir ilişkisel yapı (URL 2)*

İlişkisel veri tabanlarını kullanırken iyi bir veri tabanı mimarisi oluşturmak şarttır. Çünkü ilişkisel veri tabanlarında kurulan her bir ilişki veri tabanının performansına doğrudan etki göstermektedir ve ilişkisel veri tabanlarında veri boyutu büyüdükçe veri tabanının içerisinde oluşturulan her bir ilişkisel yapı veri tabanı performansını daha da fazla etkilemeye başlamaktadır (Boicea ve ark., 2012Jung ve ark., 2015; Öztürk ve Atmaca, 2017; Yuan ve ark., 2018).

İlişkisel veri tabanlarında büyüyen veri miktarına bağlı olarak dikey büyüme sağlanmaktadır. Dikey büyüme hem pahalıdır hem de yatay büyümeye göre daha az performans göstermektedir (Baruffa ve ark., 2019; Boicea ve ark., 2012; Hou ve ark., 2017; Putri ve Kwon, 2017). Dikey büyümede veriler bir bütün olarak aynı yerde saklanır ama bölünemez. Bu durum da maliyeti arttırır. Dikey büyümede bütün veriler bir bütün

olarak aynı yerde saklandığından dolayı veri boyutu arttıkça veri tabanı performansı kayıp yaşamaya başlar ve veri tabanının tutulduğu sunucular çok daha güçlü donanımlara ihtiyaç duymakla beraber, daha sık bakıma da ihtiyaç duyar. Bu da para ve zaman açısından ciddi kayıp anlamına gelir. İlişkisel veri tabanlarının en büyük özelliği veriler üzerinde kurulan ilişkiler ve gelişmiş sorgulama dilidir; ama bütün bunlara rağmen maliyet ve performans eksikliği bulunmaktadır (Boicea ve ark., 2012; Yuan ve ark., 2018). İlişkisel veri tabanı kullanıcılarının NoSQL (İlişkisel olmayan) veri tabanına geçmek istemelerinin nedenleri Şekil 3'te gösterilmiştir (Öztürk ve Atmaca, 2017; URL 3).



Şekil 3. *NoSQL veri tabanları tercih sebepleri (URL 3)*

İlişkisel olmayan veri tabanı

NoSQL veri tabanları yoğun okuma ve yazma işlemlerini klasik SQL (İlişkisel) veri tabanlarına kıyasla oldukça kolay ve hızlı bir şekilde yapabilmektedir (Öztürk ve Atmaca, 2017). NoSQL veri tabanları özel olarak tasarlanmıştır ve oldukça esnek yapılara sahiptir (Boicea ve

ark., 2012; Matallah ve ark., 2021; Öztürk ve Atmaca, 2017). NoSQL veri tabanları özellikle kullanım kolaylığı, esnek yapısı, performansı ve maliyeti ile popülaritesini arttırmıştır.

NoSQL yatay olarak ölçeklendirilebilen veri tabanı çözümleridir (Putri ve Kwon, 2017; Yuan ve ark., 2018). Dikey büyümede sistem kaynaklarının yetersiz kaldığı yerde sistemi yenilemek veya sistemi donanımsal olarak güncellemek gerekmektedir. Yatay büyümede ise sunucuların yetmediği durumlarda yeni sunucular alınır. İş yükleri sunucular arasında paylaştırılır. Böylece maddi ve performans açısından çok daha etkin bir çözüm elde edilmiş olur.

NoSQL veri tabanları kendi aralarında Doküman (Document), Anahtar (Key), Grafik (Graph) olarak üçe ayrılırlar (Matallah ve ark., 2021). Doküman (Document) tabanlı yapılarda veri tabanı üzerinde yapılan her kayıt bir doküman olarak nitelendirilmektedir. Yapılan kayıtlar aynı zamanda JSON formatında tutulmaktadır. MongoDB, Cassandra ve Amazon SimpleDB bunlara örnek olarak gösterilebilmektedir. Anahtar / Değer (Key / Value) tabanlı; bu tarz yapılarda anahtara karşılık gelen tek bir bilgi bulunur, kolon kavramı yoktur. Grafik (Graph) tabanlı; diğer ilişkisel olmayan veri tabanlarından farklı olarak verilerin arasındaki ilişkiyi de tutar. FlockDB buna örnek olarak gösterilebilir.

NoSQL veri tabanları SQL veri tabanlarının alternatifi olmak için ortaya çıkmış veri depolama sistemleri değildir. Aksine ilişkisel veri tabanlarının yapabildiklerini ve geneli itibari ile yapamadıklarını çok daha etkili bir şekilde ve çok da uygun maliyetlerle yapmak için ortaya çıkmış çözümlerdir. Şekil 4'te bazı NoSQL veri tabanları verilmiştir (URL 4).

Şekil 4. *Bazı NoSQL veri tabanları (URL 4)*

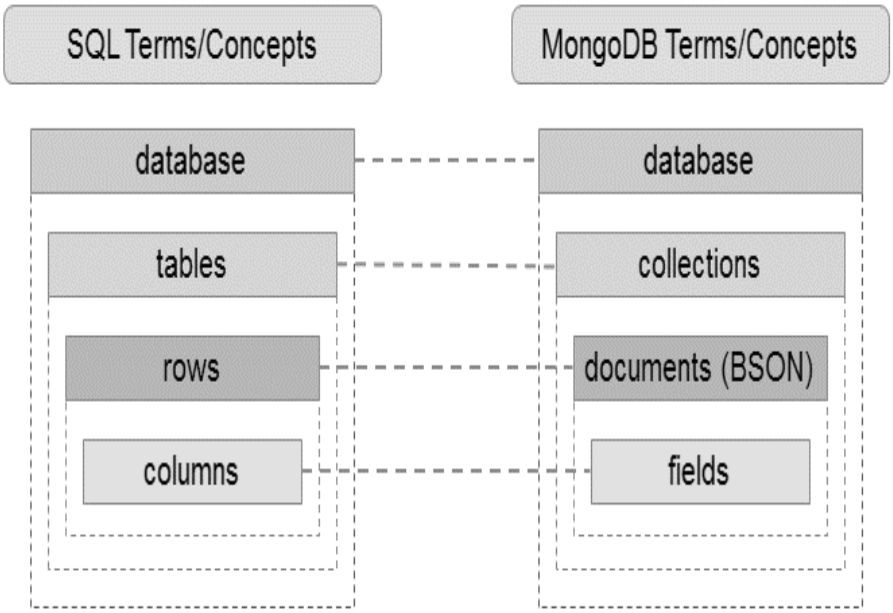
MongoDB

MongoDB açık kaynak kodlu bir ilişkisel olmayan veri tabanı uygulamasıdır. Bu çalışmada MongoDB veri tabanın seçilmesinin nedeni; MongoDB veri tabanının açık kaynak kodlu olması, sürekli güncelleme alması, dünya çapında çok geniş kullanıcı kitlesine sahip olması, geniş kütüphane desteği sunabiliyor olması, kullanım kolaylığı ve ücretsiz olmasıdır.

MongoDB, C++ programlama dili kullanılarak 2009 yılında geliştirilmiştir. MongoDB doküman tabanlı ve ölçeklenebilir bir veri tabanı uygulamasıdır. MongoDB verileri JSON\BSON biçiminde doküman olarak saklar (Baruffa ve ark., 2019; Boicea ve ark., 2012; Matallah ve ark., 2021).

MongoDB içerisinde tutulan her bir verinin kendine özgü ve eşsiz bir ID numarası vardır. MongoDB bu özgün ID numaralarını kullanarak ve gelişmiş sorgulama dili desteği ile birlikte sorgulama ve okuma ile yazma işlemlerinde özellikle ilişkisel veri tabanlarına göre oldukça başarılı performans gösterebilmektedir (Boicea ve ark., 2012; Chopade ve Dhavase, 2017).

MongoDB ilişkisel veri tabanları gibi tablo yapılarından oluşmamaktadır. MongoDB collection (Koleksiyon), document (Belge) ve field (Alan)'dan oluşmaktadır. MongoDB ve ilişkisel veri tabanları arasındaki tanımlama farklılıkları Şekil 5'te verilmiştir (Boicea ve ark., 2012; Öztürk ve Atmaca, 2017; Malik ve ark., 2020).



Şekil 5. *MongoDB ve ilişkisel veri tabanları arasındaki tanımlama farklılıkları (Malik ve ark., 2020)*

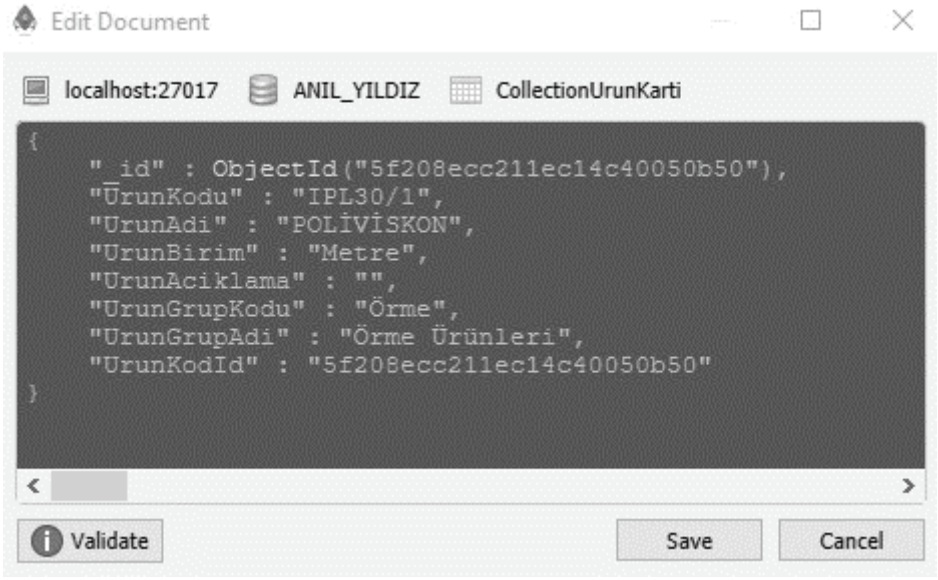
MongoDB ilişkisel veri tabanlarına göre performans konusunda bariz olarak öndedir (Patil ve ark., 2017; Rezapour ve ark., 2015).

MongoDB'nin ilişkisel yapıda kullanılması

MongoDB yapısal olarak ilişkisel olmayan bir veri tabanıdır. Ancak bu durum MongoDB'nin ilişkisel olarak kullanılamayacağı anlamına da gelmemektedir. MongoDB'yi ilişkisel olarak kullanmak için doğru bir veri tabanı tasarımı yapmak ve gelişmiş bir programlama dili kullanmak şarttır. Gelişmiş programlama dilleri ile MongoDB'nin sunmuş olduğu geniş driver (Sürücü) desteği de hesaba katıldığında geniş bir kod yazma olanağı sağlanarak kullanıcılar için çok geniş bir kullanım desteği sunulmuş olur

(Matallah ve ark., 2021; Mearaj ve ark., 2018).

MongoDB bize her bir veri kaydında eşsiz bir ID numarası verir. MongoDB'nin vermiş olduğu bu eşsiz numara, ilişkisel yapıya yakın bir yapı kurulmasına olanak sağlar. ObjectId ("5f2af47a211ec116109a03ae") verilen bu ObjectId eşsiz yapısı ile kullanıcılarına birçok şeyi yapmasına olanak sağlar (Lou ve Ye, 2018; Öztürk ve Atmaca, 2017). ObjectId'yi açıklamak gerekirse T.C kimlik numarası bunun için en uygun benzetme olur. Şekil 6'da örnek bir MongoDB verisi verirmiştir (Öztürk ve Atmaca, 2017).



Şekil 6. *Örnek MongoDB verisi*

MongoDB'nin Java, Python, C# gibi birçok programlama dili ile kullanım desteği vardır. Yapılan çalışmada C# programlama dilinin seçilmesindeki etkenler ise; geniş kullanıcı desteği, nesne tabanlı programlama dili olması, günümüzün en gelişmiş programlama dillerinden birisi olması şeklinde özetlenebilir. Şekil 7'de C# programlama dili kullanılarak veri kaydının

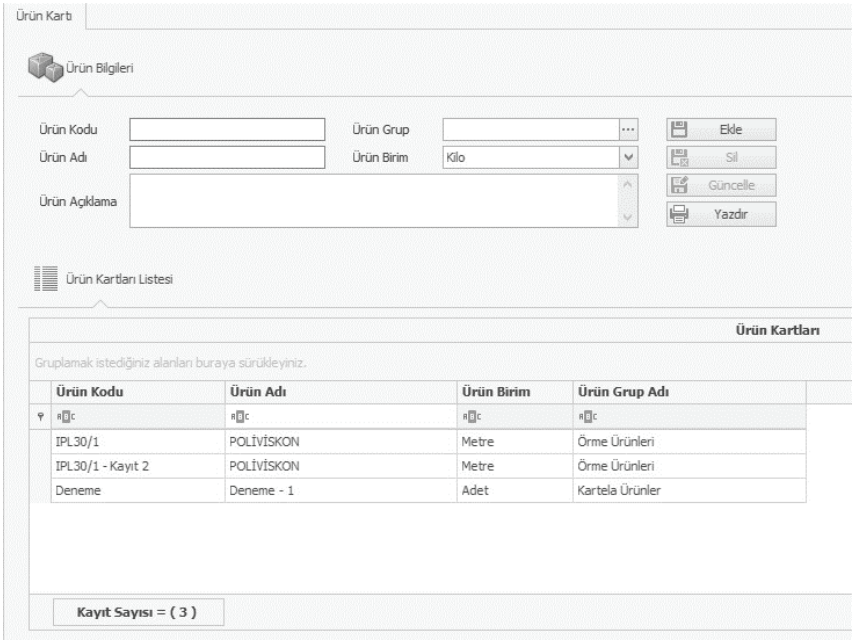
nasıl yapılabileceği görselde kodlarıyla birlikte detaylı olarak verilmiştir (Jung ve ark., 2015).

```
collection = database.GetCollection<CollectionUrunKarti>("CollectionUrunKarti");
var UrunKartlari = new CollectionUrunKarti { };

UrunKartlari.UrunKodu = txtUrunKodu.Text;
UrunKartlari.UrunAdi = txtUrunAdi.Text;
UrunKartlari.UrunBirim = txtUrunBirim.Text;
UrunKartlari.UrunGrupAdi = txtUrunGrup.Text;
UrunKartlari.UrunAciklama = txtUrunAciklama.Text;
UrunKartlari.UrunGrupKodu = UrunGrupKodu;

collection.Insert(UrunKartlari);
```

Şekil 7. *C# programlama dili kullanılarak MongoDB'ye veri kaydının yapılması.*

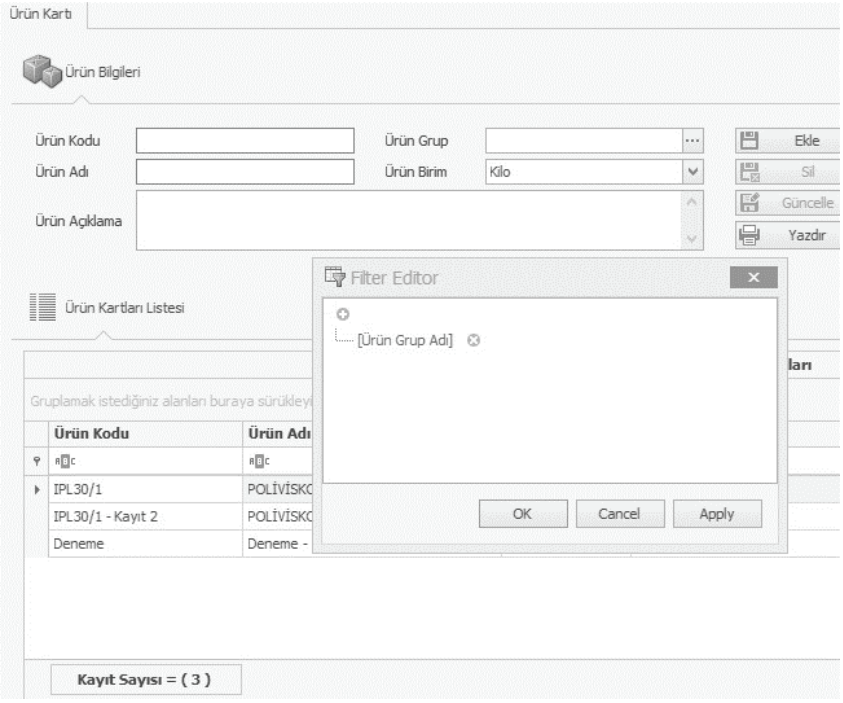
Şekil 7'de de görüldüğü üzere kullanıcı tarafından bir ObjectId parametresi girilmemektedir. ObjectId parametresi MongoDB tarafından verilen bir değerdir (Lou ve Ye, 2018; Öztürk ve Atmaca, 2017).

Microsoft visual studio üzerine ilave edilen "devexpress" eklentisi kullanıcılara gelişmiş yeni nesneler ve formlar sunar. Devexpress eklentisi ile birlikte gelen GridControl (Veri listeleme aracı) bir listeleme ve filtreleme imkânı kazandırmaktadır. Şekil 8'de GridControl üzerinde listelenmiş MongoDB verileri gösterilmiştir.



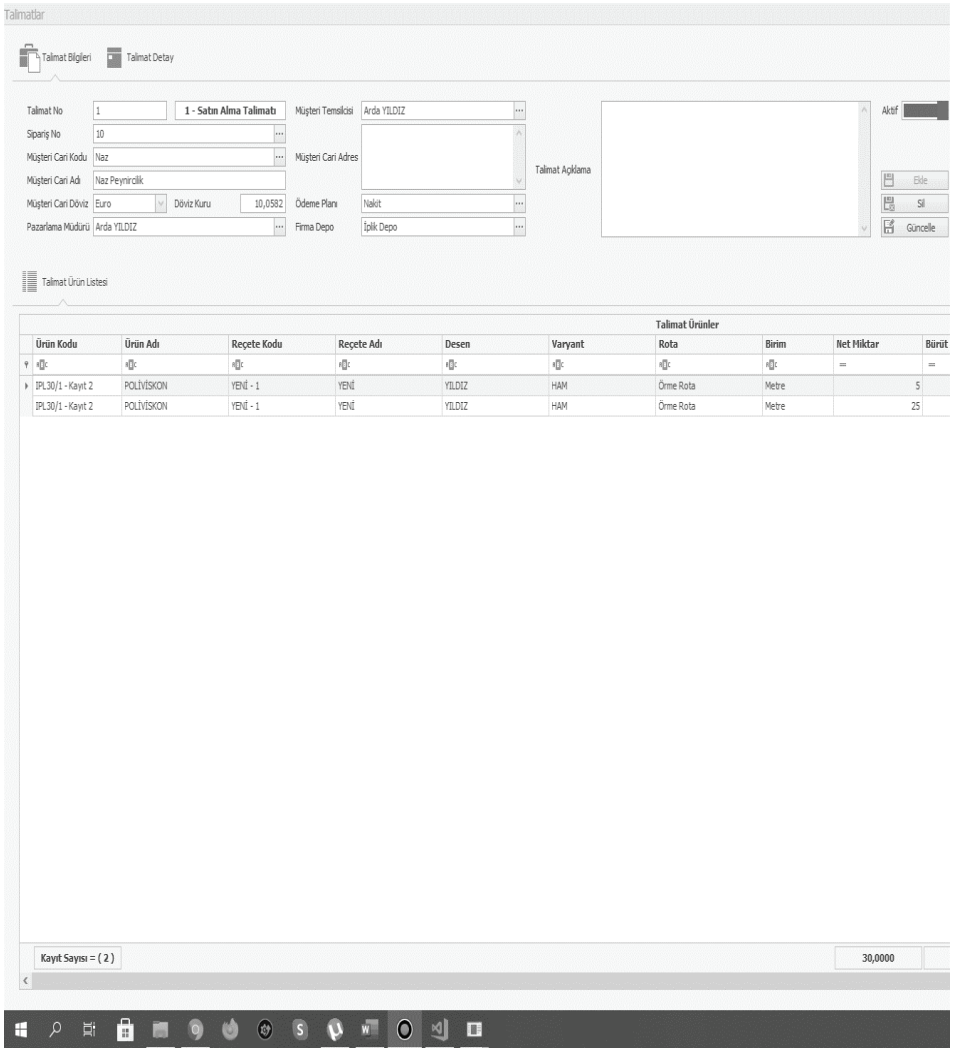
Şekil 8. *Listelenmiş MongoDB verisi*

Şekil 8 üzerinde gösterilmiş olan listeleme yöntemi ile her bir sütun için ayrı ayrı filtreleme özelliği vardır. Bunun dışında GridControl kullanıcıya kendi filtrelerini oluşturma imkanını da tanımaktadır. Şekil 9'da filtre oluşturma gösterilmiştir.



Şekil 9. *Filter editör ile filtre oluşturma*

C# programlama dili kullanılarak yapılan programın bir diğer özelliği ise sipariş, talimat, irsaliye ve fatura modüllerini içerisinde barındırması ve tıpkı SQL veri tabanlarında olduğu gibi farklı tablolar arasındaki verileri birlikte ilişkisel yapıya benzer şekilde kullanabilmesidir. Şekil 10'da talimat formu görseli verilmiştir.

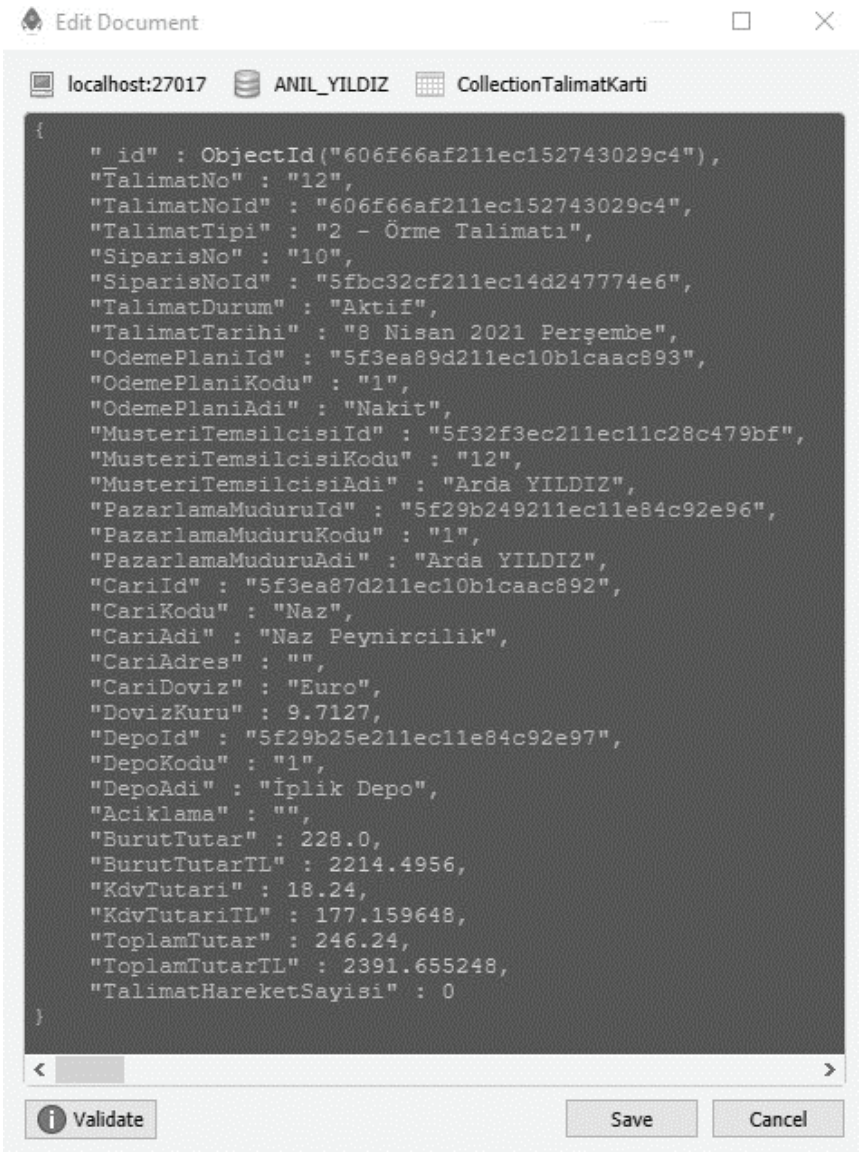


Şekil 10. *MongoDB veri tabanı kullanılarak hazırlanmış talimat formu*

Talimat formu talimat kartı ve talimat urun adlı iki farklı collection'dan oluşmaktadır. Talimat bilgileri talimat kartı collectionu içerisinde tutulmaktadır. Ama talimat kartı içerisinde tanımlanmış her bir ürün talimat urun collectionu içerisinde tutulmaktadır. Talimat formunun iki farklı collection tarafından oluşması İlişkisel veri tabanı mantığına benzemektedir. Talimat formunun içerisindeki bilgiler iki farklı collection içerisinde

tutulmasına karşın MongoDB'nin kullanıcısına sağlamış olduğu ObjectId parametresi sayesinde talimat formu içerisinde sorunsuz bir şekilde veriler arasında ilişkisel bağ kurulabilmektedir. MongoDB tarafından oluşturulan her bir ObjectId parametresi eşsiz bir değer olduğundan ötürü ilişkisel veri tabanlarında veriler arasında ilişki kurulmasını sağlayan ID parametresinin görevini üstlenebilmektedir. Talimat kartının içerisinde ekli olan her bir ürüne ait olduğu talimatın ID (TalimatNoId) numarasını kendi içerisinde taşıdığı için sadece ait olduğu talimat içerisinde listelenmektedir.

Şekil 11'de talimat kartı collectionu içerisinde bilgiler verilmiştir (Boicea ve ark., 2012; Győrödi ve ark., 2015; Matallah ve ark., 2021; Öztürk ve Atmaca, 2017).



Şekil 11. *Talimat kartı verisi*

Talimat kartı içerisinde kullanılan ürünler ile talimat kartının birbirleriyle ilişkilendirilmesi için talimat kartı içerisinde oluşturulan TalimatNoId parametresi kullanılmıştır. TalimatNoId parametresinin değeri Şekil 11 içerisinde de göründüğü üzere MongoDB tarafından oluşturulan ObjectId

parametresinin metinsel ifadeye dönüştürülmüş halidir (Chopade ve Dhavase, 2017). Şekil 12'de talimat kartı içerisinde ekli olan bir adet ürünün kaydı verilmiştir (Boicea ve ark., 2012; Győrödi ve ark., 2015; Matallah ve ark., 2021; Öztürk ve Atmaca, 2017).

Şekil 12. *Talimat kartı içerisindeki ekli oran iki ürünün kaydı*

Şekil 12'de de gözüktüğü üzere ürünler ile talimatı ilişkilendirmek için talimat içerisinde kayıtlı olan ürünlerin kayıtlarında da TalimatNoId parametresi ve değeri tutulmaktadır. Böylece artık talimat kartı ile talimat ürünleri arasında eşsiz bir ortak değere sahip olduğundan dolayı kolayca ilişki kurulabilmektedir.

Sonuç ve değerlendirme

Bu çalışma bir NoSQL veri tabanı olan MongoDB'nin ilişkisel yapıda kullanılabilmesinin mümkün olduğunu göstermesi bakımından ve günümüzde kullanılan ERP (Kurumsal kaynak planlama) programlarında da ilişkisel veri tabanları yerine MongoDB'nin kullanılabileceğini göstermesi bakımından örnek teşkil etmektedir.

Yapılan çalışma kapsamında MongoDB kullanılarak geliştirilen tekstil ERP programı sipariş, talimat, irsaliye ve faturalar gibi ilişkisel yapının kullanılmasının zorunlu olduğu modülleri başarı ile gerçekleştirebildiği görülmüştür. Bu modüllerdeki işlemlerin doğru veri tabanı tasarımı ile ilişkisel olmayan veri tabanları tarafından da başarılı bir şekilde yapılabileceğini göstermiştir.

Yapılan çalışmanın bir diğer amacı ise günümüzde kullanılan ERP programlarında veri tabanı olarak ilişkisel veri tabanlarının tercih edilmesinden ötürü doğan performans, sürdürebilirlik ve maliyet gibi sorunlara çözüm bulmak olup, bulunan çözümün ise bir ERP uygulaması ile desteklenmesi amaçlanmıştır.

NoSQL ilişkisel veri tabanlarına göre okuma ve yazma hızı konusunda oldukça öndedir. İlişkisel olmayan veri tabanları ayrıca yatay mimariye sahip olduklarından sürdürülebilirlik ve maliyet konusunda da ilişkisel veri tabanlarına göre oldukça avantajlıdır (Boicea ve ark., 2012; Győrödi ve ark., 2015; Prabagaren, 2014).

Bu çalışmada MongoDB'nin kullanılmasının en önemli nedeni ise programcıya geniş kullanım desteği sağlaması ve günümüzün en gelişmiş ilişkisel olmayan veri tabanlarından birisi olmasıdır (Öztürk ve Atmaca, 2017).

Kaynaklar

[1] Baruffa, G., Femminella, M., Pergolesi, M., & Reali, G. (2019). Comparison of MongoDB and Cassandra databases for spectrum monitoring As-a-Service. *IEEE Transactions on Network and Service Management*, *17*(1), 346-360.

[2] Boicea, A., Radulescu, F., & Agapin, L. I. (2012). MongoDB vs Oracle--database comparison. In *2012 third international conference on emerging intelligent data and web technologies* (pp. 330-335). IEEE.

[3] Chopade, M. R. M., ve Dhavase, N. S. (2017). Mongodb, couchbase: Performance comparison for image dataset. In *2017 2nd International Conference for Convergence in Technology* (I2CT) (pp. 255-258). IEEE.

[4] Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015). A comparative study: MongoDB vs. MySQL. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1-6). IEEE.

[5] Hou, B., Shi, Y., Qian, K., ve Tao, L. (2017). Towards analyzing mongodb noSQL security and designing injection defense solution. In *2017 IEEE 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)* (pp. 90-95). IEEE.

[6] Jung, M. G., Youn, S. A., Bae, J., ve Choi, Y. L. (2015). A study on data input and output performance comparison of mongodb and postgreSQL in the big data environment. In *2015 8th International Conference on Database Theory and Application (DTA)* (pp. 14-17). IEEE.

[7] Lou, Y., ve Ye, F. (2018). Research on Data Query Optimization Based

on SparkSQL and MongoDB. In *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)* (pp. 144-147). IEEE.

[8] Mearaj, I., Maheshwari, P., ve Kaur, M. J. (2018). Data Conversion from Traditional Relational Database to MongoDB using XAMPP and NoSQL. In *2018 Fifth HCT Information Technology Trends (ITT)* (pp. 94-98). IEEE.

[9] Öztürk, S., ve Atmaca, H. E. (2017). İlişkisel ve ilişkisel olmayan (NoSQL) veri tabanı sistemleri mimari performansının yönetim bilişim sistemleri kapsamında incelenmesi. *Bilişim Teknolojileri Dergisi*, *10*(2), 199-209.

[10] Patil, M. M., Hanni, A., Tejeshwar, C. H., ve Patil, P. (2017). A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing—Sharding in MongoDB and its advantages. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (pp. 325-330). IEEE.

[11] Prabagaren, G. (2014). Systematic approach for validating Java-MongoDB schema. In *International Conference on Information Communication and Embedded Systems (ICICES2014)* (pp. 1-4). IEEE.

[12] Putri, F. K., ve Kwon, J. (2017). A distributed system for finding high profit areas over big taxi trip data with MognoDB and spark. In *2017 IEEE International Congress on Big Data (BigData Congress)* (pp. 533-536). IEEE.

[13] Rezapour, M., Moradi, M., ve Ghadiri, N. (2015). Performance evaluation of SQL and MongoDB databases for big e-commerce data. In *2015 International Symposium on Computer Science and Software Engineering (CSSE)* (pp. 1-7). IEEE.

[14] Yuan, M., Liu, K., Zhang, L., ve Zou, C. (2018). Research on Big Data Storage Model of Oilfield Assay Data Based on MongoDB. In *2018 IEEE 4th International Conference on Computer and Communications*

(ICCC) (pp. 1863-1866). IEEE.

[15] Matallah, H., Belalem, G., & Bouamrane, K. (2021). Comparative Study Between the MySQL Relational Database and the MongoDB NoSQL Database. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *13*(3), 38-63.

[16] Malik, A., Burney, A., & Ahmed, F. (2020). A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems. *Journal of Computer and Communications*, *8* (4), 59-71.

İnternet Kaynakları

URL 1 - https://koraypeker.com/2019/03/16/modern-veri-tabanlari (Erişim Tarihi: 08.04.2021)

URL 2 - https://www.red-gate.com/ (Erişim Tarihi: 15.04.2021)

URL 3 - https://www.turkz.org/ (Erişim Tarihi: 21.04.2021)

URL 4 - http://chadwickspencer.com/myblog/index.php/2018/04/ (Erişim Tarihi: 27.03.21)