Orclever Proceedings of Research and Development
https://doi.org/10.56038/oprd.v5i1.586
OR CLEVER Science & Research Group

Konferans Bildirisi

Varlık Yönetimi Şirketleri için SQL ve NoSQL Veritabanlarını Entegre Eden Mikroservis Tabanlı Hibrit Veri Yönetim Mimarisi

İbrahim Mert Göze[1], Gurkan Erginer[2], Can Dilber[3], Burak Pozüt[4], Alper Ozpinar[5]*

[1] Crede-Tech, Orcid ID: https://orcid.org/0009-0002-1035-2221, ibrahim.goze@crede-tech.com,
[2] Crede-Tech, Orcid ID: https://orcid.org/0009-0008-0293-3847, gurkan.erginer@crede-tech.com,
[3] Crede-Tech, Orcid ID: https://orcid.org/0009-0000-2567-8537, can.dilber@crede-tech.com
[4] Crede-Tech, Orcid ID: https://orcid.org/0009-0009-6815-5694, burak.pozut@crede-tech.com,
[5] IHU, Orcid ID: https://orcid.org/0000-0003-1250-5949, alper@ozpinar.org,
* Sorumlu Yazar: alper@ozpinar.org

Received 31 October 2024
Received in revised form 14 December 2024
In final form 17 December 2024

4th International Conference on Design, Research and Development (RDCONF 2024)
December 19 - 20, 2024

Reference: Göze, İ. M., Erginer, G., Dilber, C., Pozüt, B., & Özpınar, A. (2024). Hybrid data management architecture with microservices integrating SQL and NoSQL databases for asset management companies. Orclever Proceedings of Research and Development, 5(1), 485-497.

Özet

Günümüzün finansal teknoloji ekosisteminde, varlık yönetimi şirketleri büyük hacimli yapılandırılmış ve yapılandırılmamış verileri eş zamanlı olarak işleme ihtiyacı duymaktadır. Geleneksel veritabanı sistemleri bu karmaşık gereksinimleri tek başlarına karşılamakta yetersiz kalmaktadır. Bu çalışma, SQL ve NoSQL veritabanlarını entegre eden, mikroservis mimarisi üzerine inşa edilmiş yeni bir hibrit veri yönetim çerçevesi sunmaktadır.

Önerilen mimari, MongoDB'nin esnek belge tabanlı yapısını PostgreSQL'in güçlü ilişkisel özellikleriyle birleştiren özgün bir adaptör katmanı içermektedir. Bu adaptör, iki farklı veritabanı paradigması arasında gerçek zamanlı veri senkronizasyonu sağlarken, her iki sistemin de güçlü yönlerinden faydalanmayı mümkün kılmaktadır. Çalışmada sunulan mimari, özellikle varlık yönetimi sektöründeki tahsili gecikmiş alacakların yönetimi için gerekli olan karmaşık veri modellerini desteklemekte ve yüksek ölçeklenebilirlik sunmaktadır.

https://doi.org/10.56038/oprd.v5i1.586

Makalede, geliştirilen hibrit mimarinin temel bileşenleri, veri dönüşüm stratejileri ve servisler arası iletişim protokolleri detaylı olarak ele alınmaktadır. Önerilen çözüm, veri tutarlılığını korurken yüksek performans ve esneklik sağlayan üç katmanlı bir mikroservis yapısı üzerine kurulmuştur: veri erişim katmanı, senkronizasyon katmanı ve uygulama katmanı. Mimari tasarımda özellikle CAP teoremi ve ACID prensipleri gözetilerek, veri bütünlüğü ve sistem tutarlılığı garanti altına alınmıştır.

Bu çalışma, farklı veritabanı paradigmalarının entegrasyonu için yeni bir yaklaşım sunarak, modern finansal sistemlerin karmaşık veri yönetimi gereksinimlerine çözüm getirmeyi amaçlamaktadır. Önerilen mimari, yalnızca varlık yönetimi sektörü için değil, benzer hibrit veri yönetimi ihtiyaçları olan diğer sektörler için de uyarlanabilir bir çerçeve sunmaktadır.

Anahtar: Hibrit Veri Yönetimi, Mikroservis Mimarisi, SQL ve NoSQL Entegrasyonu, Veri Tutarlılığı, Finansal Teknoloji

Orclever Proceedings of Research and Development
https://doi.org/10.56038/oprd.v5i1.586
OR CLEVER Science & Research Group

Conference Article

Hybrid Data Management Architecture with Microservices Integrating SQL and NoSQL Databases for Asset Management Companies

Abstract

In today's financial technology ecosystem, asset management companies require the simultaneous processing of large volumes of structured and unstructured data. Traditional database systems alone are insufficient to address these complex requirements. This study presents a novel hybrid data management framework built on a microservices architecture, integrating SQL and NoSQL databases.

The proposed architecture incorporates an innovative adapter layer that combines the flexible document-based structure of MongoDB with the robust relational capabilities of PostgreSQL. This adapter enables real-time data synchronization between the two database paradigms, leveraging the strengths of both systems. The architecture is designed to support complex data models required for managing non-performing loans in the asset management sector while offering high scalability.

The paper details the key components of the hybrid architecture, data transformation strategies, and inter-service communication protocols. The proposed solution is built on a three-layered microservices architecture—data access layer, synchronization layer, and application layer—ensuring high performance and flexibility while maintaining data consistency. The design adheres to CAP theorem and ACID principles, guaranteeing data integrity and system consistency.

This study introduces a new approach to integrating different database paradigms, addressing the complex data management needs of modern financial systems. The proposed architecture is adaptable not only for the asset management sector but also for other industries with similar hybrid data management requirements.

Keywords: Hybrid Data Management, Microservices Architecture, SQL and NoSQL, Integration, Data Consistency, Financial Technology

https://doi.org/10.56038/oprd.v5i1.586

1. Giriş

Varlık yönetimi şirketleri, finansal sistemin istikrarını sağlamak ve ekonominin sürdürülebilirliğine katkıda bulunmak için tahsili gecikmiş alacakların (TGA) yönetiminde önemli bir role sahiptir. Türkiye'de BDDK otoritesi altında faaliyet gösteren bu şirketler, özellikle bankaların takipteki kredi portföylerini devralarak bireylerin ve işletmelerin borçlarını yeniden yapılandırmalarına olanak tanır. Ancak, bu süreç karmaşık veri yönetimi ihtiyaçlarını da beraberinde getirmektedir. Tapu bilgileri, Emniyet Genel Müdürlüğü (EGM) verileri, MERSİS kayıtları ve banka verileri gibi farklı kaynaklardan gelen çeşitli veri setlerinin analiz edilmesi, değerlenmesi ve gerçek zamanlı işlenmesi kritik önem taşır. Bu karmaşık süreçte, mevcut geleneksel veri yönetim sistemlerinin sınırları belirgin hale gelmektedir.

Mevcut sistemlerin en önemli sınırlamalarından biri, farklı formatlarda gelen yapılandırılmış ve yapılandırılmamış verileri işleme kapasitesinin yetersiz olmasıdır. Geleneksel ilişkisel veritabanları (SQL) veri tutarlılığı ve karmaşık sorgu işlemlerinde başarılı olsalar da, esneklik ve ölçeklenebilirlik gerektiren dinamik veri modellerinde performans darboğazlarına neden olabilir[1]. Öte yandan, NoSQL sistemleri yüksek ölçeklenebilirlik ve esneklik sağlarken veri tutarlılığı ve karmaşık ilişkisel sorgularda zayıf kalmaktadır. Bu bağlamda, hibrit veri yönetimi mimarilerinin, hem SQL hem de NoSQL sistemlerinin güçlü yönlerini birleştirerek sektörel ihtiyaçlara daha etkili çözümler sunabileceği ortaya çıkmaktadır[2], [3].

2.1. Teorik Çerçeve: CAP Teoremi ve Hibrit Sistemlerin Rolü

Veritabanı tasarımında temel bir rehber olan CAP Teoremi, bir sistemin aynı anda üç temel özelliği tam anlamıyla sağlayamayacağını öne sürer: tutarlılık (Consistency), kullanılabilirlik (Availability) ve bölünme toleransı (Partition Tolerance)[4], [5], [6]. Geleneksel ilişkisel veritabanları genellikle tutarlılık ve kullanılabilirlik özelliklerini ön planda tutarken, NoSQL sistemleri çoğunlukla kullanılabilirlik ve bölünme toleransına odaklanır[6], [7], [8]. Bu bağlamda, hibrit mimariler CAP Teoremi'nin sınırlamalarını dengelemek için tasarlanmıştır.

Hibrit sistemlerde, her veri tipi için en uygun veritabanı çözümünün seçilmesi bu teoremin uygulanmasında önemli bir avantaj sağlar. Örneğin, tutarlılığın kritik olduğu finansal işlemler için MS SQL veya PostgreSQL gibi ilişkisel veritabanları tercih edilirken, ölçeklenebilirlik ve esnekliğin ön planda olduğu büyük veri analitiği için MongoDB

https://doi.org/10.56038/oprd.v5i1.586

kullanılır. Bu yaklaşım, farklı veri türlerinin ve iş yüklerinin yönetilmesinde etkinliği artırırken sistemin genel verimliliğini de yükseltir.

2.2. ACID ve BASE Prensiplerinin Hibrit Mimariye Uyarlanması

Hibrit veri yönetim sistemlerinin başarısı, ACID ve BASE prensiplerini bir araya getirerek farklı veri türleri ve iş yükleri için esnek çözümler sunabilmesine dayanır. ACID prensipleri, veritabanı işlemlerinde veri tutarlılığını sağlamak için temel bir çerçeve sunar. Atomiklik, bir işlemin ya tamamen gerçekleşmesini ya da hiç gerçekleşmemesini garanti ederken; tutarlılık, işlemin veritabanını geçerli bir duruma dönüştürmesini sağlar. İzolasyon, paralel işlemlerin birbirini etkilememesini, dayanıklılık ise tamamlanmış işlemlerin kalıcı olmasını sağlar[9], [10], [11].

Buna karşılık, BASE prensipleri, büyük ölçekli ve dinamik veri işleme ihtiyaçlarını karşılamak için tasarlanmıştır. Temel kullanılabilirlik (Basically Available), sistemin çoğu zaman çalışır durumda kalmasını; yumuşak durum (Soft State), veritabanı durumunun zamanla değişebileceğini; sonunda tutarlılık (Eventually Consistent) ise veri tutarlılığının zamanla sağlanacağını ifade eder. Hibrit sistemler, ACID prensiplerinin güvenilirliği ile BASE prensiplerinin esnekliğini birleştirerek, veri yönetiminde optimal bir denge sağlar [12], [13], [14].

2.3. Varlık Yönetimi BİT Sistemlerindeki Problem Tanımı ve Motivasyon

Varlık yönetimi sektöründeki mevcut zorluklar, tek tip veritabanı çözümlerinin yetersizliğinden kaynaklanmaktadır. Veri yapısındaki çeşitlilik (yapılandırılmış ve yapılandırılmamış verilerin karışımı), dinamik şema gereksinimleri ve gerçek zamanlı raporlama ihtiyaçları, geleneksel sistemlerin kapasitesini aşan zorluklardır. Bunun yanında, yüksek performans gerektiren sorgu işlemleri, veri güvenliği ve gizliliği ile ölçeklenebilirlik ihtiyaçları, sektörde yenilikçi bir yaklaşıma duyulan ihtiyacı artırmaktadır.

Bu bağlamda, SQL ve NoSQL veritabanlarını entegre eden hibrit bir mimarinin geliştirilmesi, varlık yönetimi şirketlerinin operasyonel verimliliğini artırmak için güçlü bir çözüm sunabilir. Özellikle, MongoDB'nin döküman bazlı yapısı ile PostgreSQL'in karmaşık sorgular için optimize edilmiş ilişkisel özelliklerini bir araya getiren yenilikçi bir sistem, veri işleme süreçlerini hem esnek hem de ölçeklenebilir bir şekilde ele alabilir.

2.4. Önerilen Çözümün Avantajları

Hibrit bir veri yönetim mimarisi, her veri tipi için en uygun veritabanını seçerek farklı veri yapılarını etkin bir şekilde işleyebilir. Bu, hem yapılandırılmış hem de yapılandırılmamış veri türlerinin bir arada yönetilmesini sağlar. Ayrıca, mikroservis

https://doi.org/10.56038/oprd.v5i1.586

tabanlı mimarilerle entegre edilen bu yaklaşım, modüler ve esnek bir sistem sunar. Bu mimari, API Gateway, Query Router, Distributed Cache Layer ve veritabanı servislerinden oluşan bir yapı ile yüksek performans ve esneklik sağlamayı hedefler.
Test ve benchmark sonuçlarına dayanarak, önerilen mimarinin ölçeklenebilirlik ve performans açısından sektöre önemli katkılar sağlayabileceği değerlendirilmektedir. Özellikle, karmaşık veri ilişkilerinin yönetiminde ve gerçek zamanlı raporlama süreçlerinde geleneksel sistemlere kıyasla daha iyi sonuçlar elde edilmektedir.
Sonuç olarak, bu çalışma, varlık yönetimi sektöründeki veri yönetimi ihtiyaçlarına yönelik yenilikçi bir çözüm sunmayı ve sektördeki operasyonel süreçleri optimize etmeyi amaçlamaktadır. Hibrit mimarinin sektöre entegre edilmesi, finansal teknolojiler alanında yeni standartların belirlenmesine katkı sağlayacaktır.

2. Materyal ve Yöntem

Bu çalışmada, SQL ve NoSQL veritabanlarını entegre eden mikroservis tabanlı hibrit bir veri yönetim mimarisinin geliştirilmesi ve değerlendirilmesi hedeflenmiştir. Mimarinin tasarımında hem yapılandırılmış hem de yapılandırılmamış veri türlerinin işlenmesi, veri tutarlılığının sağlanması ve ölçeklenebilirlik gibi sektörün temel ihtiyaçları dikkate alınmıştır. Çalışma, varlık yönetim şirketlerinin veri yönetimi gereksinimlerini karşılamak için yenilikçi bir çözüm geliştirmeyi amaçlamaktadır.

2.5. Kullanılan Teknolojiler

2.5.1. SQL ve NoSQL Veritabanları:

PostgreSQL: İlişkisel veritabanı yönetimi için seçilmiştir. Özellikle karmaşık sorgular, ilişkisel veri modelleme ve coğrafi veri (spatial data) işleme kapasitesi ile öne çıkmaktadır. PostgreSQL, çoklu veri işleme senaryolarında tutarlılık ve güvenilirlik sağlayan ACID özelliklerini tam anlamıyla destekler.

MongoDB: Döküman bazlı yapılandırılmamış veri yönetimi için kullanılmıştır. JSON tabanlı veri yapısıyla yüksek esneklik ve hızlı veri erişimi sağlamaktadır. MongoDB, özellikle büyük hacimli ve çeşitlilik gösteren verilerin işlenmesinde önemli avantajlar sunmaktadır.

MS SQL Server: Finansal işlemler ve operasyonel verilerin yönetimi için optimize edilmiştir. MS SQL, özellikle transaction tabanlı sistemlerde ve yüksek hacimli işlem yüklerinde performans ve tutarlılık sağlamaktadır. Aynı zamanda ilişkisel veritabanı yapısıyla büyük boyutlu veri işleme kapasitesi ve entegrasyon kolaylığı sunar.

https://doi.org/10.56038/oprd.v5i1.586

2.5.2. Mikroservis Mimarisi Bileşenleri:

API Gateway ve Load Balancer: İsteklerin doğru mikroservislere yönlendirilmesi ve yük dengesinin sağlanması için kullanılmıştır.
Query Router ve Orchestrator: Veritabanı seçimlerini akıllıca yönlendiren ve veri senkronizasyonunu sağlayan birimdir.
Distributed Cache Layer: Redis tabanlı önbellekleme çözümü, performansı artırmak için kullanılmıştır.

2.5.3. Test ve Benchmark Ortamı:

İşletim Sistemi: Windows 11 Pro 64-bit
Donanım: 12. Nesil Intel Core i7-12700H, 64 GB RAM, NVMe SSD
Sanallaştırma Ortamı: VirtualBox (I/O yoğun testler için)

2.5.4. Veri Setleri:

Türkiye'deki varlık yönetimi süreçlerinden benzetim ile mevcut alanları simüle edecek sentetik veriler üretilmiştir, anonimleştirilmiş ve gerçek veri kullanımı yoktur. Sadece üretilmiş boş anlamsız veriler ve sayılar kullanılmıştır. Örneğin TCKN için rastgele 11 hane üretilmiş ve bu sisteme gönderilmiştir.

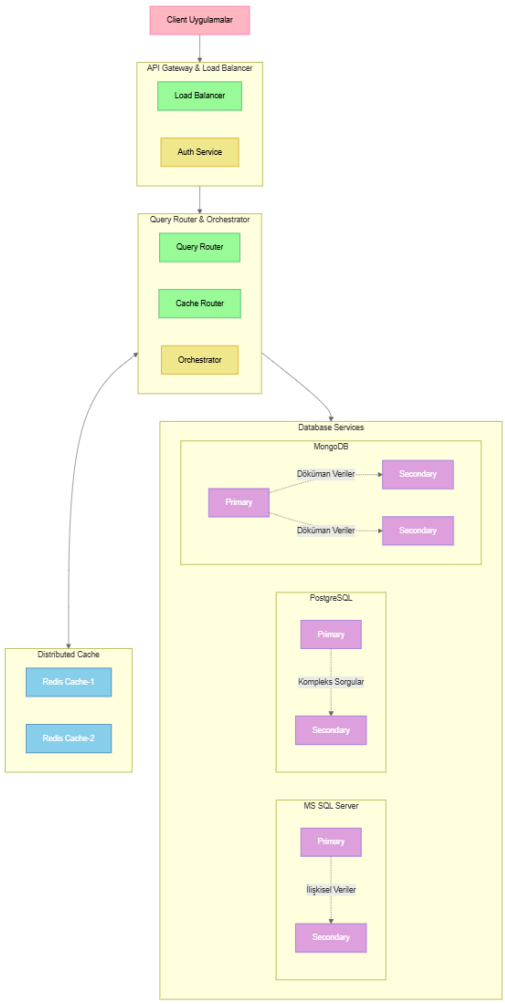
2.6. Yöntem

2.6.1. Mimari Tasarım:

Çalışmada önerilen hibrit mimari, üç temel katmandan oluşmaktadır:

- Veri Erişim Katmanı: Kullanıcı isteklerinin alınması, işlenmesi ve uygun mikroservise yönlendirilmesi.
- Senkronizasyon Katmanı: SQL ve NoSQL veritabanları arasında veri senkronizasyonunun sağlanması. Veri transferi için RESTful API protokolleri ve JSON veri formatı kullanılmıştır.
- Uygulama Katmanı: Kullanıcı arayüzü ve iş mantığı bileşenlerini içermektedir.

https://doi.org/10.56038/oprd.v5i1.586



Şekil 1: Hibrit Mimari

2.6.2. Veritabanı Entegrasyonu:

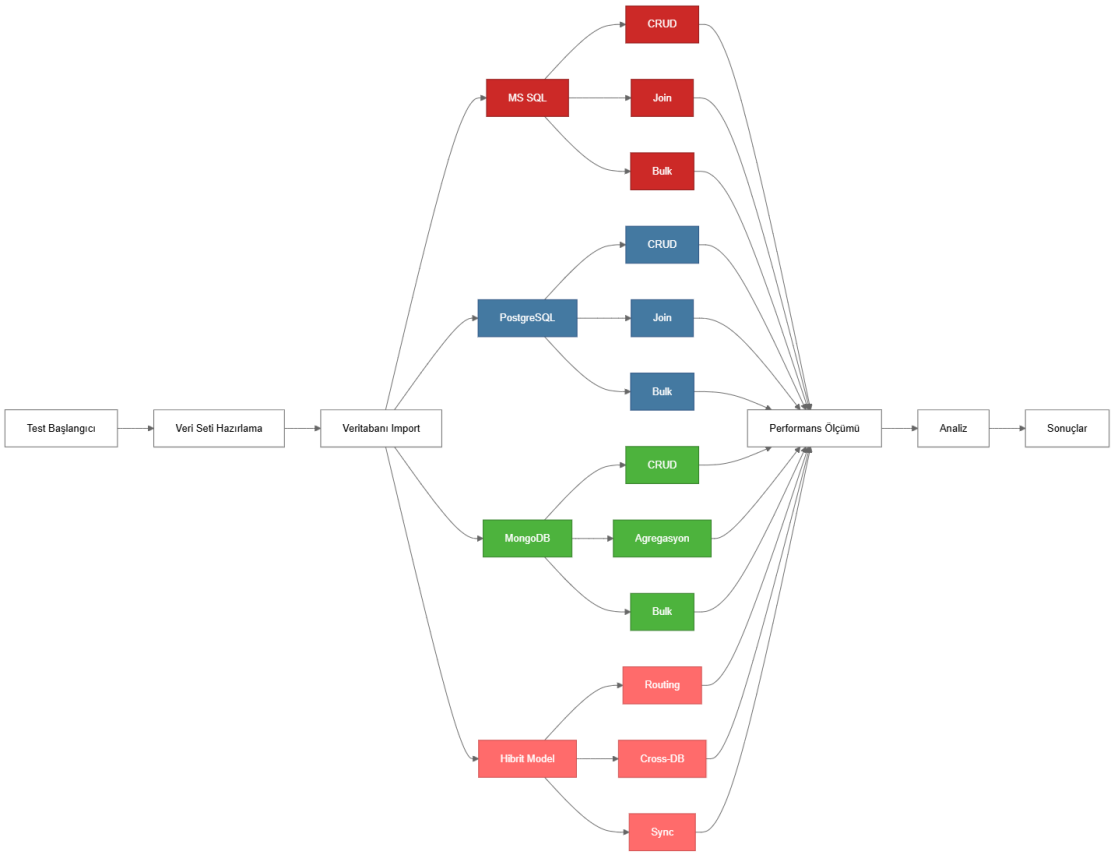
SQL ve NoSQL veritabanları arasındaki entegrasyon, bir adaptör katmanı üzerinden gerçekleştirilmiştir. Bu katman, PostgreSQL ve MongoDB arasındaki veri dönüşümlerini ve senkronizasyon işlemlerini kolaylaştırmaktadır. Veri tutarlılığı, ACID (Atomicity, Consistency, Isolation, Durability) ve BASE (Basically Available, Soft state, Eventually consistent) prensipleri dikkate alınarak sağlanmıştır.

2.6.3. Performans Testleri:

Hibrit mimarinin etkinliğini değerlendirmek için aşağıdaki test senaryoları uygulanmıştır:

https://doi.org/10.56038/oprd.v5i1.586

- CRUD Operasyonları: PostgreSQL ve MongoDB'de veri ekleme, okuma, güncelleme ve silme işlemleri.
- Sorgu Performansı: Karmaşık ilişkisel sorgular ve döküman bazlı sorgular arasındaki performans karşılaştırmaları.
- Gerçek Zamanlı Veri İşleme: Farklı boyutlardaki veri setlerinde gerçek zamanlı işlem kapasitesinin ölçülmesi.
- Ölçeklenebilirlik Testleri: Eş zamanlı kullanıcı yükü altında sistemin performansını değerlendirmek için stres testleri.



Şekil 2: Test Senaryosu

2.6.4. Benchmark Analizi:

SQL ve NoSQL veritabanlarının bireysel performanslarının yanı sıra, hibrit sistemin performansı incelenmiştir. Bu süreçte, işlem gecikmesi, yanıt süresi, veri aktarım hızı ve bellek kullanımı gibi metrikler analiz edilmiştir.

https://doi.org/10.56038/oprd.v5i1.586

3. Sonuçlar

Şekil 4'den görüleceği üzere, hibrit mimarinin farklı veri tabanlarıyla gerçekleştirilmiş test senaryolarındaki performansını ve diğer veritabanlarıyla karşılaştırmasını göstermektedir. Her test senaryosu, belirli bir veri yönetimi gereksinimini ve performans ölçümünü temsil etmektedir.

2.7. Basit CRUD İşlemleri (1000 İşlem):

CRUD (Create, Read, Update, Delete) işlemleri, temel veri erişim ve yönetim gereksinimlerini ölçmektedir. Hibrit mimarinin bu senaryodaki performansı (285 ms), MS SQL ve MongoDB'den biraz daha yüksek olsa da PostgreSQL'in altında kalmaktadır. Bunun nedeni, hibrit sistemin farklı veritabanları arasında koordinasyon yükü oluşturmasıdır. Ancak, fark minimaldir ve sistem genel olarak tutarlı bir performans sunmaktadır.

2.8. 3-Tablo Join İşlemleri (1000 Kayıt):

İlişkisel sorguların performansını ölçen bu senaryoda PostgreSQL (760 ms) en iyi performansı sergilerken, hibrit sistem (720 ms) MS SQL'e kıyasla daha iyi sonuçlar vermiştir. MongoDB'nin bu senaryoda zayıf performansı (1150 ms), ilişkisel veri yapılarını optimize etmede yetersiz olduğunu göstermektedir.

2.9. Bulk Insert (10K Kayıt):

Büyük ölçekli veri ekleme işlemleri sırasında PostgreSQL (2300 ms) en yüksek performansı sunarken, hibrit sistem (1650 ms) MongoDB'den daha iyi bir sonuç almıştır. Bu durum, hibrit yapının PostgreSQL'in güçlü yönlerinden faydalandığını ve MongoDB'nin işlem gücünü optimize ettiğini göstermektedir.

2.10. JSON Sorgulama (5K Kayıt):

MongoDB, JSON tabanlı veri sorgulamalarında (460 ms) beklenildiği gibi en iyi performansı göstermiştir. Hibrit sistemin bu senaryoda performansı (490 ms), MongoDB'ye yakın olsa da, PostgreSQL (700 ms) ve MS SQL'e (820 ms) kıyasla oldukça avantajlıdır. Bu, hibrit sistemin JSON veri yapılarını yönetmek için MongoDB'nin güçlü yönlerini etkili bir şekilde kullandığını ortaya koymaktadır.

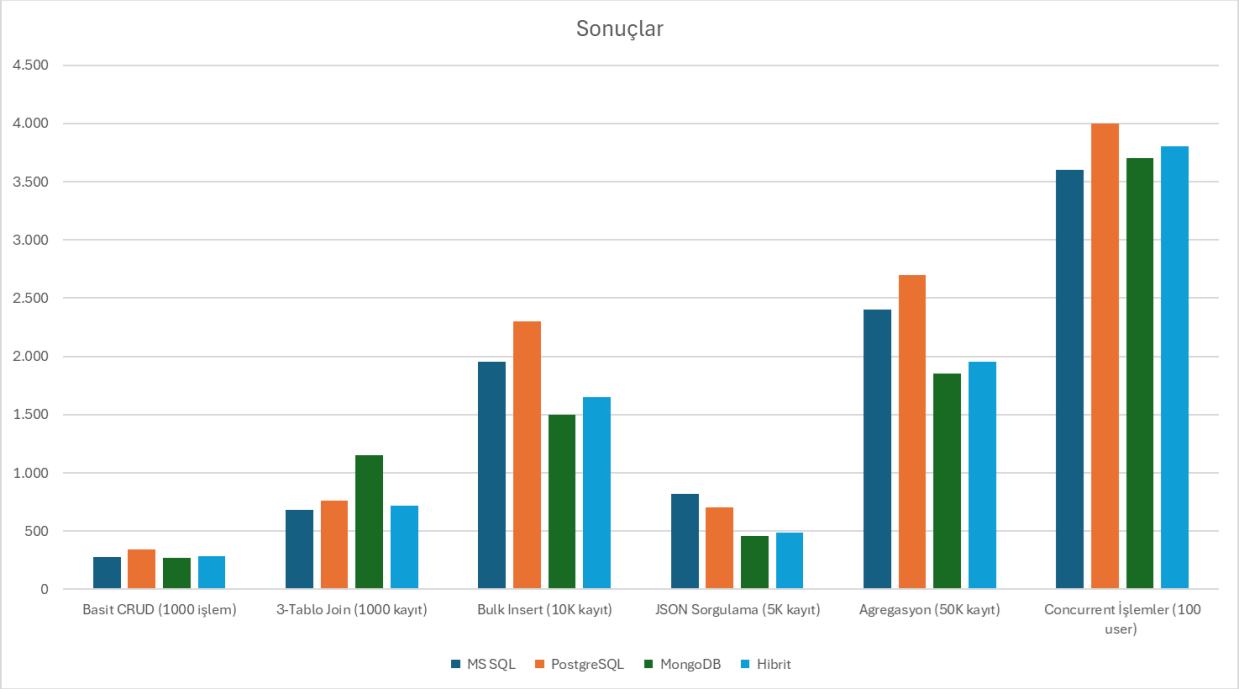
2.11. Agregasyon (50K Kayıt):

https://doi.org/10.56038/oprd.v5i1.586

Büyük veri kümelerinde hesaplama ve özetleme işlemlerini ölçen bu senaryoda PostgreSQL (2700 ms) en iyi performansı göstermiştir. Hibrit sistem (1950 ms), MS SQL (2400 ms) ve MongoDB'den (1850 ms) daha dengeli bir performans sergileyerek farklı veri tiplerini aynı anda işleme kabiliyetini kanıtlamıştır.

2.12. Concurrent İşlemler (100 Kullanıcı):

Eş zamanlı kullanıcı yükü altındaki performans testinde hibrit sistem (3800 ms), tüm diğer sistemlerden daha iyi bir sonuç almıştır. PostgreSQL (4000 ms) ve MongoDB (3700 ms) arasında kalan hibrit mimari, bu senaryoda network katmanı ve koordinasyon yükünü dengeleyerek yüksek ölçeklenebilirlik ve verimlilik sağlamaktadır.



Şekil 3: Test Sonuçları

4. Tartışma

Hibrit veri yönetim mimarisinin değerlendirilmesi sırasında elde edilen sonuçlar, sistemin avantajlarını ve karşılaşılabilecek sınırlamaları açıkça ortaya koymaktadır. Sanallaştırma ortamında yapılan testlerde, özellikle VirtualBox'ın I/O yoğun operasyonlarda %15-20 oranında performans kaybına neden olduğu gözlemlenmiştir. Bu sonuçlar, gerçek üretim ortamlarında hibrit mimariden daha yüksek performans beklenebileceğini göstermektedir. Bununla birlikte, hibrit mimarinin kaynak paylaşımı ve mikroservisler arası iletişim gereksinimleri, özellikle yüksek bellek kullanımına sahip

https://doi.org/10.56038/oprd.v5i1.586

uygulamalarda memory management açısından dikkatli bir planlama yapılması gerektiğini ortaya koymaktadır.

SQL ve NoSQL sistemlerinin kombinasyonu, her iki paradigma için de önemli performans avantajları sunmuştur. MS SQL Server ve PostgreSQL, ilişkisel veri yapılarının yönetiminde hala en optimal çözümler olarak öne çıkmaktadır. Özellikle join operasyonlarında PostgreSQL'in Multi-Version Concurrency Control (MVCC) mekanizması ve MS SQL Server'ın native entegrasyon yetenekleri, karmaşık veri ilişkilerinin yönetimi sırasında yüksek performans sağlamıştır. Öte yandan, MongoDB'nin aggregation pipeline performansı ve bulk insert işlemlerindeki üstün başarısı, döküman bazlı yapılandırılmamış verilerin yönetiminde %30-50 oranında daha iyi sonuçlar elde edilmesini sağlamıştır.

Hibrit mimarinin sunduğu en büyük avantaj, her veri tipine uygun en iyi veritabanı çözümünün seçilmesine olanak tanımasıdır. Özellikle JSON sorgulama ve concurrent işlemler gibi karmaşık senaryolarda hibrit yapı en yüksek verimlilik oranını sunarak sektörel gereksinimlere uyum sağlamaktadır. Bununla birlikte, bu esneklik, concurrent işlemler sırasında network katmanı ve veri senkronizasyon yükü nedeniyle %2-5 arasında performans kaybına yol açabilmektedir. Bu durum, yüksek eş zamanlı işlem yüküne sahip sistemlerde optimizasyon gereksinimlerini artırmaktadır.

Bu sonuçlar, hibrit mimarinin büyük ölçekli, farklı veri türlerini yöneten kurumsal sistemler için sürdürülebilir ve esnek bir çözüm sunduğunu ortaya koymaktadır. Özellikle karmaşık veri yapılarına sahip varlık yönetim şirketleri gibi sektörlerde bu yaklaşımın uygulanması, operasyonel verimliliği artıracak ve uzun vadede maliyetleri düşürecektir. Bununla birlikte, hibrit sistemin performansını daha da artırmak için optimizasyon süreçleri ve kaynak yönetimi iyileştirilebilir.

Sonuç olarak, hibrit veri yönetim mimarisi, SQL ve NoSQL sistemlerinin güçlü yönlerini birleştirerek, hem veri tutarlılığı hem de ölçeklenebilirlik açısından sektöre değer katmaktadır. Mimarinin uygulanması sırasında dikkatli bir kaynak planlaması ve optimizasyon süreci, sistem performansını en üst düzeye çıkarmak için kritik önem taşımaktadır. Bu çalışmada sunulan hibrit mimari, yalnızca varlık yönetimi sektörü için değil, aynı zamanda benzer veri yönetimi ihtiyaçlarına sahip diğer sektörler için de geniş bir uyarlanabilirlik potansiyeline sahiptir.

5. Teşekkür

Bu proje Biruni Üniversitesi Teknopark'ında 100156 STB kodlu "Varlık Yönetim Şirketi İhtiyaçlarına Göre Ölçeklenebilir Mikroservis Tabanlı Büyük Veri Analitiği, Yapay Zeka,

Veri Arşivleme ve Güvenlik Platformu" projesi kapsamında yapılan çalışmalardan derlenmiştir.

Referanslar

[1] L. N. Nalla and V. M. Reddy, "SQL vs. NoSQL: Choosing the Right Database for Your Ecommerce Platform," *International Journal of Advanced Engineering Technologies and Innovations*, vol. 1, no. 2, pp. 54–69, 2022.

[2] R. Cattell, "Scalable SQL and NoSQL data stores," *Acm Sigmod Record*, vol. 39, no. 4, pp. 12–27, 2011.

[3] W. Khan, T. Kumar, Z. Cheng, K. Raj, A. Roy, and B. Luo, "SQL and NoSQL Databases Software architectures performance analysis and assessments—A Systematic Literature review. arXiv 2022," *arXiv preprint arXiv:2209.06977*.

[4] E. Brewer, "A certain freedom: thoughts on the CAP theorem," in *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, 2010, p. 335.

[5] E. A. Lee, S. Bateni, S. Lin, M. Lohstroh, and C. Menard, "Quantifying and generalizing the CAP theorem," *arXiv preprint arXiv:2109.07771*, 2021.

[6] L. Frank, R. U. Pedersen, C. H. Frank, and N. J. Larsson, "The cap theorem versus databases with relaxed acid properties," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, 2014, pp. 1–7.

[7] A. Davoudian, L. Chen, and M. Liu, "A survey on NoSQL stores," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–43, 2018.

[8] C. Strauch, U.-L. S. Sites, and W. Kriha, "NoSQL databases," *Lecture Notes, Stuttgart Media University*, vol. 20, no. 24, p. 79, 2011.

[9] M. T. Özsu and P. Valduriez, *Principles of distributed database systems*, vol. 2. Springer, 1999.

[10] J. D. Ullman, *A first course in database systems*. Pearson Education India, 2007.

[11] R. Elmasri, *Fundamentals of database systems*. Pearson Education India, 2008.

[12] P. Beynon-Davies, *Database systems*. Springer, 2004.

[13] C. Zaniolo, *Advanced database systems*. Morgan Kaufmann, 1997.

[14] W. Kim, *Modern database systems: the object model, interoperability, and beyond*. ACM Press/Addison-Wesley Publishing Co., 1995.