Melike Orhan

Öğr No:02230224001

İLİŞKİSEL VE İLİŞKİSEL OLMAYAN VERİ TABANI SİSTEMLERİ MİMARİ PERFORMANSININ YÖNETİM BİLİŞİM SİSTEMLERİ KAPSAMINDA İNCELENMESİ

İlişkisel ve ilişkisel olmayan (NoSQL) veritabanı sistemleri, modern bilişim sistemlerinin temelini oluşturan iki farklı veri yönetimi yaklaşımını temsil eder. Her biri, farklı mimari özelliklere, veri modellerine ve kullanım senaryolarına sahiptir. Bu sistemlerin performansının yönetim bilişim sistemleri kapsamında incelenmesi, birçok önemli konuyu içerir.

1. **Performans Ölçütleri ve Metrikleri**: İlişkisel ve ilişkisel olmayan veritabanı sistemlerinin performansını ölçmek için kullanılan metrikler ve yöntemlerin belirlenmesi önemlidir. Bu metrikler, veritabanı erişim hızı, işlem işleme süresi, yüksek erişilebilirlik, veri bütünlüğü gibi faktörleri içerebilir.
2. **Veri Modeli ve Mimarilerin Karşılaştırılması**: İlişkisel ve NoSQL veritabanları, farklı veri modelleri ve mimarilere sahiptir. İlişkisel veritabanları genellikle tablo-tabanlı bir yapı kullanırken, NoSQL veritabanları genellikle belge, grafik, anahtar-değer vb. veri modellerini destekler. Bu farklılıkların performansa etkisi incelenmelidir.
3. **Dağıtılmış Sistemler ve Yük Dengeleme**: NoSQL veritabanları genellikle dağıtılmış sistemler olarak tasarlanır ve yatay olarak ölçeklenebilirler. Bu, yüksek trafikli uygulamalarda performans avantajları sağlayabilir. İlişkisel veritabanlarında bu tür dağıtılmış mimariler daha karmaşıktır ve genellikle dikey olarak ölçeklenir. Bu bağlamda, dağıtılmış sistemlerin performans yönetimi büyük önem taşır.
4. **Veri Yoğun Uygulamalar ve Büyük Veri**: NoSQL veritabanları, büyük veri hacimlerini ve hızlı veri akışlarını işlemek için daha iyi donanımlı olabilir. Özellikle IoT (Nesnelerin İnterneti) gibi veri yoğun uygulamalarda performans önemlidir. İlişkisel veritabanları, genellikle yapısal verilere daha iyi uygunluk sağlar, ancak büyük veri işleme gereksinimleri karşısında bazı zorluklarla karşılaşabilir.
5. **Yedekleme ve Kurtarma Stratejileri**: Performans yönetimi sadece hızlı veri işleme ile ilgili değildir, aynı zamanda veri bütünlüğü, yedekleme ve kurtarma stratejilerini de içerir. Hem ilişkisel hem de NoSQL veritabanlarında bu stratejilerin etkin bir şekilde yönetilmesi gereklidir.
6. **İş Yükü Tahminleri ve Kapasite Planlaması**: Performans yönetimi aynı zamanda gelecekteki iş yüklerini tahmin etme ve buna göre altyapıyı ölçeklendirme yeteneğini de içerir. İş yükü tahminleri ve kapasite planlaması, veritabanı sistemlerinin performansını optimize etmede önemli bir rol oynar.

Sonuç olarak, ilişkisel ve ilişkisel olmayan veritabanı sistemlerinin mimari performansının yönetim bilişim sistemleri kapsamında incelenmesi, farklı veri yönetimi stratejilerini, mimarilerini ve performans etkileyen faktörleri anlamak için önemlidir. Bu inceleme, organizasyonların veri tabanı çözümlerini seçerken ve uygularken daha bilinçli kararlar almalarına yardımcı olabilir.

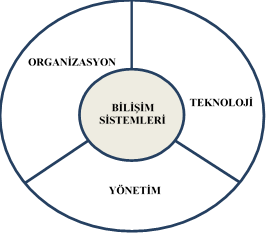
İlişkisel ve İlişkisel Olmayanların İncelenmesi (NoSQL) Veritabanı Sisteminin Mimari Performansları Bilgi Sistemleri Yönetim Şartları

Bu başlık altında, ilişkisel ve ilişkisel olmayan (NoSQL) veritabanı sistemlerinin incelenmesi, NoSQL veritabanı sistemlerinin mimari performansları ve bilgi sistemleri yönetim şartlarına ilişkin genel bir bakış sunabiliriz.

1. **İlişkisel ve İlişkisel Olmayan Veritabanı Sistemlerinin İncelenmesi**:
   * İlişkisel Veritabanı Sistemleri: İlişkisel veritabanı sistemleri, verileri tablolar halinde saklar ve SQL (Structured Query Language) kullanarak verilere erişimi sağlar. Veri bütünlüğü için ACID (Atomiklik, Tutarlılık, İzole Edilebilirlik, Dayanıklılık) özelliklerine dayanır.
   * İlişkisel Olmayan (NoSQL) Veritabanı Sistemleri: İlişkisel olmayan veritabanı sistemleri, geleneksel tablo yapılarına bağlı kalmadan veri depolama ve sorgulama sağlar. Esnek veri modelleri, yüksek ölçeklenebilirlik ve dağıtık mimariler gibi avantajlar sunabilir. NoSQL sistemleri, SQL dili yerine çeşitli sorgulama dilleri veya API'ler kullanır.
2. **NoSQL Veritabanı Sisteminin Mimari Performansları**:
   * Yüksek Ölçeklenebilirlik: NoSQL veritabanı sistemleri, dağıtık mimariler ve yatay ölçeklenebilirlik özellikleri sayesinde büyük miktarda veriyi işleme ve depolama kapasitesine sahip olabilir.
   * Esnek Veri Modelleri: NoSQL sistemleri, belirli bir şema (schema) ile sınırlı kalmadan çeşitli veri modellerini destekler. Bu, veri yapısındaki değişikliklere daha iyi uyum sağlar.
   * Yüksek Performans: NoSQL sistemleri, özellikle büyük veri hacimlerinde ve yüksek hızlı veri yazma ve okuma operasyonlarında yüksek performans sağlayabilir.
3. **Bilgi Sistemleri Yönetim Şartları**:
   * Veri Bütünlüğü ve Güvenlik: Hem ilişkisel hem de NoSQL veritabanı sistemlerinde, veri bütünlüğü ve güvenliği sağlamak önemlidir. Veriye yetkisiz erişimleri engellemek ve veri bütünlüğünü korumak için uygun güvenlik önlemleri alınmalıdır.
   * Yedekleme ve Kurtarma: Her iki sistemde de veri yedekleme ve kurtarma stratejileri önemlidir. Veri kaybını önlemek ve hızlı bir şekilde sistemleri geri yüklemek için düzenli yedeklemeler yapılmalı ve kurtarma planları hazırlanmalıdır.
   * Performans İzleme ve Optimizasyon: Bilgi sistemleri yöneticileri, hem ilişkisel hem de NoSQL veritabanı sistemlerinin performansını izlemeli ve gerektiğinde optimizasyonlar yapmalıdır. Bu, veritabanı performansının maksimum düzeyde tutulmasını sağlar.

Bu faktörler, hem ilişkisel hem de ilişkisel olmayan veritabanı sistemlerinin etkili bir şekilde yönetilmesi için önemlidir ve bu sistemlerin performanslarının ve güvenilirliğinin sağlanmasına yardımcı olur.

# BİLİŞİM SİSTEMLERİ VE YÖNETİMİ

Bilgi sistemleri ve yönetimi, bir organizasyonun bilgi işleme ihtiyaçlarını karşılamak için bilgisayar ve iletişim teknolojilerini kullanarak bilgiyi toplama, depolama, işleme, yönetme ve dağıtma sürecini içerir.

Formun Üstü

Formun Üstü

bilişim sistemleri yönetiminin ana bileşenleri:

1. **Stratejik Planlama:** Bilişim sistemleri yönetimi, organizasyonun genel iş stratejileriyle uyumlu olarak BT stratejilerinin belirlenmesini içerir. Bu stratejiler, teknoloji yatırımlarının yönlendirilmesi, BT altyapısının geliştirilmesi ve işletme ihtiyaçlarına uygun BT hizmetlerinin sağlanmasını içerir.
2. **BT Altyapısı Yönetimi:** BT altyapısı, ağlar, sunucular, depolama sistemleri, veritabanları ve diğer donanım ve yazılım bileşenlerini içerir. Bilişim sistemleri yönetimi, bu altyapının planlanması, uygulanması, güncellenmesi ve bakımının sağlanmasını içerir. Ayrıca BT altyapısının güvenliği ve bütünlüğü de önemli bir konudur.
3. **Proje Yönetimi:** BT projelerinin planlanması, uygulanması, izlenmesi ve tamamlanması bilişim sistemleri yönetimi kapsamındadır. Projeler genellikle yeni yazılım uygulamalarının geliştirilmesi, altyapı güncellemeleri, sistem entegrasyonları veya dijital dönüşüm projeleri gibi alanlarda gerçekleşir.
4. **BT Hizmet Yönetimi:** BT hizmet yönetimi, organizasyonun BT hizmetlerinin etkili bir şekilde sağlanmasını ve işletilmesini içerir. Bu, kullanıcı destek hizmetleri, sistem bakımı, güvenlik yönetimi, performans izleme ve raporlama gibi alanları kapsar. Hizmet seviyesi anlaşmaları (SLA'lar) genellikle bu sürecin bir parçasıdır.
5. **Bilgi Güvenliği Yönetimi:** Bilgi güvenliği, organizasyonun bilgi varlıklarını korumak için alınan önlemleri içerir. Bilişim sistemleri yönetimi, güvenlik politikalarının belirlenmesi, güvenlik açıklarının tespiti ve düzeltilmesi, güvenlik eğitimlerinin sağlanması ve kriz durumlarına karşı hazırlıklı olunması gibi alanlarda bilgi güvenliği yönetimini içerir.
6. **BT Performans Yönetimi:** Bilişim sistemleri yönetimi, BT sistemlerinin ve hizmetlerinin performansını izler, değerlendirir ve iyileştirme fırsatlarını tanımlar. Bu, sistemlerin verimli bir şekilde çalışmasını sağlamak için gereken kapasite planlaması, performans izleme ve optimizasyonu içerir.
7. **Yasal ve Uyum Yönetimi:** Organizasyonların faaliyet gösterdiği sektörlere ve bölgelere bağlı olarak, bilişim sistemleri yönetimi, ilgili yasal düzenlemelere ve uyumluluk gereksinimlerine uygunluğu sağlar. Veri koruma yönetmelikleri, veri gizliliği kuralları ve diğer sektörel düzenlemelere uyum sağlamak bu sürecin bir parçasıdır.

Bilişim sistemleri yönetimi, organizasyonların rekabet avantajı elde etmelerine ve iş hedeflerine ulaşmalarına yardımcı olan kritik bir işlevdir. Her organizasyonun, kendi ihtiyaçlarına ve hedeflerine uygun olarak bu süreci etkin bir şekilde yönetmesi önemlidir.

Formun Üstü

VERİ TABANI VE VERİ TABANI YÖNETİM SİSTEMLERİ :Veri tabanı en genel tanımıyla, kullanım amacına uygun olarak düzenlenmiş veriler topluluğudur. Birbirleriyle ilişkileri olan verilerin tutulduğu, mantıksal ve fiziksel olarak tanımlarının olduğu bilgi depolarıdır. Veri tabanları gerçekte var olan ve birbirleriyle ilişkisi olan nesneleri ve ilişkileri modeller . Veri tabanı yönetimsistemleri (VTYS), verilere aynı anda birden çok bağlantı sağlayabilme özelliği sağlar. Bu sistemler, veri tabanı yönetiminin bir parçası olarak, verinin nasıl depolanacağı, kullanılacağı ve erişileceğini mantıksal olarak yönlendiren bir kurallar sistemidir.

Veri tabanı modellerini yedi kategoriye ayırabiliriz:

1)Düz model veya tablo modeli: İki boyutlu veri grubundan oluşur. Sütunlarda verilerin benzer özellikleri, satırlarda ise veri grupları yer alır. Kullanıcı adlarının ve şifrelerinin tutulduğu veri tabanı buna örnek olarak verilebilir. Böyle bir veri tabanında her satırda bir kullanıcıya ait şifre bilgileri, sütunlarda ise tipleri aynı olan veriler yer alır. Düz veri modeli tek tablodan oluşan bir model olarak düşünülebilir.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ad Soyad** | **Kullanıcı Adı** | **Parola** |
| **Kayıt 1** | Murat ERGİN | Mergin | kjVdb125 |
| **Kayıt 2** | Ayşe YILMAZ | Ayılmaz | Bks46db7 |

2)Hiyerarşik Veri Modeli : Hiyerarşik veritabanı, verilerin bir ağaç yapısı şeklinde organize edildiği ve ilişkilendirildiği bir veritabanı türüdür. Bu tür veritabanlarında veriler, üst düzeyden alt düzeye doğru bir hiyerarşi içinde düzenlenirler. Her veri öğesi, yalnızca bir üst düzey öğeyle ilişkilendirilir ve alt düzeydeki öğeler bu üst düzey öğelerle ilişkilendirilir.

Hiyerarşik veritabanlarının en temel bileşeni, kayıtların düğümler tarafından temsil edildiği bir ağaç yapısıdır. Bu düğümler genellikle bir anahtar-satır şeklinde düşünülür. Bir anahtar-satır, birden çok alt düzey kaydı içerebilir, ancak her biri yalnızca bir üst düzey anahtar-satıra bağlıdır. Hiyerarşik veri modelinin temel özellikleri şunlardır:

**Düğümler (Nodes)**: Hiyerarşik veri modelinde her düğüm, bir veri öğesini temsil eder. Bu düğümler, alt düğümlerle (alt veri öğeleri) ilişki içindedir ve genellikle üst düğüme (üst veri öğesi) bağlıdır.

**Üst-Alçak İlişkiler (Parent-Child Relationships)**: Her düğüm, bir üst düğümle (parent) ve bir veya daha fazla alt düğümle (child) ilişkilidir. Bu ilişkiler, bir üst düğümün altında bulunan düğümleri belirtir.

**Kök Düğüm (Root Node)**: Hiyerarşik ağacın en üst düzeyinde bulunan ve diğer tüm düğümlerin üstünde yer alan düğümdür. Bu düğüm, hiyerarşik yapıdaki başlangıç noktasını temsil eder.

**Alt Düğümler (Child Nodes)**: Bir düğümün altında bulunan ve bu düğüme bağlı olan diğer düğümlerdir. Alt düğümler, bir üst düğüme bağlıdır ve bu üst düğüme doğrudan bağlıdır.

**Veri Yapısının Sabit Olması**: Hiyerarşik veri modeli, genellikle belirli bir veri yapısına sahiptir ve bu yapı genellikle oluşturulduktan sonra değiştirilemez. Yani, veri yapısının değişmesi genellikle zordur.

Hiyerarşik veri modeli genellikle ağaç yapısı şeklinde düzenlenen verilerin temsilinde kullanılır. Özellikle dosya sistemleri, organizasyon yapıları, klasör ve dosya yapıları gibi alanlarda yaygın olarak kullanılır. Örneğin, XML (eXtensible Markup Language) belgeleri, bir hiyerarşik veri modeli kullanır. Oracle gibi bazı veritabanı sistemleri de hiyerarşik veri modellerini destekler.

Formun Üstü

3) Ağ Veri Modeli :   
Ağ veri modeli, veritabanı yönetiminde kullanılan bir veri modelidir. İlişkisel olmayan (non-relational) bir veri modeli olarak bilinir ve hiyerarşik veri modeline bir alternatif olarak geliştirilmiştir. Ağ veri modelinde veriler, kayıtlar ve bunların arasındaki ilişkiler bir ağ yapısı içinde temsil edilir. Bu model, kayıtların birbirine bağlı olduğu ve her bir kaydın birden fazla kayıta bağlanabileceği karmaşık ilişkileri yönetmek için kullanılır. Ağ veri modelinin temel özellikleri şunlardır:

**Düğümler (Nodes)**: Her bir düğüm, bir veri öğesini temsil eder. Örneğin, bir öğrenci, bir müşteri veya bir ürün gibi.

**Kenarlar (Edges)**: Düğümler arasındaki ilişkileri temsil eder. Bu ilişkiler, iki düğüm arasında doğrudan bir bağlantı olabilir. Örneğin, bir öğrencinin bir dersi alması gibi.

**Yönlendirilmiş Kenarlar**: Bazı ağ modellerinde kenarlar yönlendirilmiş olabilir. Bu durumda, bir kenarın başlangıç ve bitiş noktaları belirlenir ve bir düğümden diğerine doğru bir yönlendirme söz konusudur. Örneğin, bir öğrencinin bir dersi aldığını ifade eden bir yönlendirilmiş kenar, öğrenci düğümünden ders düğümüne doğru yönlendirilmiş olabilir.

**Düğümler Arasındaki İlişkilerin Çeşitliliği**: Ağ veri modelinde, düğümler arasındaki ilişkiler farklı tiplerde olabilir ve bu ilişkilerin karmaşıklığı çeşitli durumlara göre değişebilir.

**Karmaşıklık ve Esneklik**: Ağ veri modeli, çeşitli veri yapıları ve ilişkileri temsil edebilme yeteneği ile karmaşıklık ve esneklik sağlar. Bu, farklı veri yapılarının ve ilişkilerin etkili bir şekilde yönetilebilmesini sağlar.

Ağ veri modelleri, genellikle graf teorisi kavramlarına dayanır ve karmaşık ilişkileri ve yapıları temsil etmek için kullanılır. Özellikle sosyal ağ analizi, iletişim ağları, ağ haritalama ve lojistik gibi alanlarda ağ veri modelleri kullanılır. Bazı popüler ağ veri tabanları arasında Neo4j ve ArangoDB bulunmaktadır.

Formun Üstü

4)İlişkisel Veri Modeli : İlişkisel veri modeli, veritabanı yönetiminde yaygın olarak kullanılan bir veri modelidir. Bu modelde, veriler tablolar şeklinde düzenlenir ve tablolar arasındaki ilişkiler belirli kurallar çerçevesinde tanımlanır. İlişkisel veri modelinin temel bileşenleri şunlardır:

**Tablolar (Tables)**: Verilerin düzenlendiği temel yapı birimleridir. Her tablo, belirli bir konsepti temsil eder ve tablodaki her bir satır, bir veri öğesini (kayıt) temsil ederken, sütunlar ise bu öğelerin özelliklerini (alanları) tanımlar.

**Satırlar (Rows)**: Bir tablodaki her bir satır, bir veri kaydını temsil eder. Her satır, tablodaki sütunlara karşılık gelen veri değerlerini içerir.

**Sütunlar (Columns)**: Bir tablodaki her bir sütun, belirli bir türde veri öğesinin (alanın) değerlerini içerir. Sütunlar, genellikle bir veri tipi (örneğin, tamsayı, metin, tarih) ve gerekirse bir sınırlama veya kısıtlama belirtir.

**Anahtarlar (Keys)**: İlişkisel veri modelinde, bir tablodaki satırları benzersiz bir şekilde tanımlayan bir veya daha fazla sütun bulunur. Bu sütunlara bir anahtar denir. Bir tabloda genellikle birincil anahtar (primary key) olarak belirtilen bir anahtar bulunur ve bir veya daha fazla sütunun birleşimiyle oluşur. Ayrıca, tablolar arasındaki ilişkileri kurmak için kullanılan yabancı anahtarlar (foreign keys) da bulunabilir.

**İlişkiler (Relationships)**: İlişkisel veri modelinde, farklı tablolar arasında ilişkiler tanımlanabilir. Bu ilişkiler, bir tablodaki anahtar sütunların, diğer tablodaki anahtar sütunlarla eşleştiği durumları ifade eder.

İlişkisel veri modeli, karmaşık veri yapılarını temsil etmek ve veri manipülasyonunu gerçekleştirmek için SQL (Structured Query Language) gibi bir sorgulama dili kullanır. İlişkisel veritabanı sistemleri, bu modelin temelinde çalışır ve kullanıcıların veritabanlarına veri ekleme, güncelleme, sorgulama ve silme gibi işlemleri gerçekleştirmelerine olanak tanır. Önde gelen ilişkisel veritabanı sistemleri arasında Oracle, MySQL, PostgreSQL, Microsoft SQL Server ve SQLite bulunmaktadır.

5)Nesne İlişkisel Veri Modeli : Nesne İlişkisel Veri Modeli, veritabanı tasarımında daha karmaşık ilişkisel yapıları tanımlamak ve bu yapıları nesne yönelimli programlama dilinde kullanmak için kullanışlıdır. Bu model, veritabanı uygulamalarının daha esnek, genişletilebilir ve nesne yönelimli programlama dilleriyle daha uyumlu olmasını sağlar. Nesne İlişkisel Veri Modelinin temel özellikleri şunlardır:

**Nesne Yönelimli Programlama Özellikleri**: Nesne İlişkisel Veri Modeli, veritabanında varlıkları (nesneleri) ve bu varlıklar arasındaki ilişkileri (ilişkileri) nesne yönelimli programlama kavramlarına benzer şekilde modelleyebilme yeteneği sağlar. Bu, veritabanı tasarımında daha yüksek düzeyde soyutlama ve modülerlik sağlar.

**Veri Kapsülleme ve Soyutlama**: Veri ve işlevsellik, nesne yönelimli programlama dilinde olduğu gibi nesnelerin içine kapsüllenir. Bu, veriye doğrudan erişimi sınırlayarak daha güvenli ve tutarlı bir veri yönetimi sağlar.

**Kalıtım ve Polimorfizm**: Nesne İlişkisel Veri Modeli, kalıtım ve polimorfizm gibi nesne yönelimli programlama kavramlarını destekler. Bu, veritabanı tasarımında daha fazla esneklik ve tekrar kullanılabilirlik sağlar.

**SQL Uzantıları**: Nesne İlişkisel Veri Modeli, SQL diline nesne yönelimli programlama özelliklerini ekleyen özel SQL uzantılarını destekler. Bu, karmaşık sorguları ve işlemleri gerçekleştirmek için daha güçlü bir veri sorgulama aracı sağlar.

**Performans ve Ölçeklenebilirlik**: Nesne İlişkisel Veri Modeli, ilişkisel veritabanı sistemlerinin performans ve ölçeklenebilirlik avantajlarından yararlanır. Bu, büyük veri hacimlerini etkin bir şekilde yönetme ve işleme yeteneği sunar.

Nesne İlişkisel Veri Modeli, özellikle karmaşık veri yapılarını ve ilişkileri daha doğru bir şekilde modellemek ve yönetmek gerektiğinde kullanışlıdır. Örneğin, finansal uygulamalar, müşteri ilişkileri yönetimi (CRM) sistemleri ve coğrafi bilgi sistemleri (GIS) gibi alanlarda bu model sıklıkla kullanılır. Önde gelen ilişkisel veritabanı sistemleri arasında Oracle, PostgreSQL ve Microsoft SQL Server, nesne ilişkisel veri modelini destekler.

Formun Üstü

Bu da yazılım geliştirme sürecinde daha verimli ve etkili bir şekilde çalışmayı mümkün kılar.

6) Çoklu Ortam Veri Modeli: Çoklu Ortam Veri Modeli (Multimedia Data Model), birden fazla ortam türünü (örneğin metin, ses, görüntü, video) desteklemek üzere tasarlanmış bir veri modelidir. Geleneksel ilişkisel veritabanı modelleri genellikle metin tabanlı verilere odaklanırken, çoklu ortam veri modelleri, görsel, işitsel ve diğer medya türlerini de içerecek şekilde genişletilmiştir.

Bu veri modeli, multimedya verilerinin saklanması, yönetilmesi, sorgulanması ve erişilmesi için özel veri tipleri ve ilişkilendirme teknikleri sunar. Örneğin, bir çoklu ortam veri modeli, ses dosyaları için özel ses veri tipleri ve görüntüler için özel görüntü veri tipleri gibi özelliklere sahip olabilir.

Çoklu ortam veri modellerinin temel özellikleri şunlardır:

**Medya Çeşitliliği**: Metin, resim, ses, video gibi farklı medya türlerini temsil etmek ve işlemek için uygun yapıları içerir.

**Zengin İçerik Yönetimi**: Çoklu ortam veri modelleri, medya içeriğinin depolanması, sorgulanması, alınması ve işlenmesi için uygun araçlar sağlar.

**Birlikte İşleme**: Metin, resim, ses ve video gibi farklı medya türlerini birlikte işlemek için olanak sağlar. Örneğin, bir video içinde konuşulanları transkribe etmek veya bir resimdeki nesneleri tanımak gibi.

**Veri Bütünlüğü ve Tutarlılık**: Çoklu ortam veri modelleri, medya içeriğinin bütünlüğünü korumak ve veri tutarlılığını sağlamak için geliştirilmiş yöntemler sunar.

**Ölçeklenebilirlik**: Büyük medya dosyaları ve yüksek hacimli medya verilerini etkin bir şekilde işlemek için ölçeklenebilir mimariler sağlar.

**Gelişmiş İşlevsellik**: Görüntü tanıma, konuşma tanıma, duygusal analiz gibi gelişmiş işlevler sağlayarak, medya içeriğinin derinlemesine analizini ve yorumlanmasını mümkün kılar.

Çoklu ortam veri modelleri, eğlence endüstrisi, dijital pazarlama, medya yayıncılığı, sağlık sektörü (örneğin, medikal görüntüleme), güvenlik ve gözetim gibi birçok alanda kullanılır. Bu tür veri modellerinin temsilinde, genellikle JSON (JavaScript Object Notation), XML (eXtensible Markup Language), RDF (Resource Description Framework) gibi yapılar kullanılır ve özel medya dosyaları için de uygun kodlama standartları kullanılır (örneğin, JPEG, PNG, MP3, MP4 vb.).

Formun Üstü

* 7) Dağıtık Veri Modeli: Dağıtık veri tabanları, iki ya da daha fazla bilgisayarda depolanan ve bir ağ üzerinde dağıtılan bilgiler için kullanılan veri tabanı grubudur. Veri tabanını ağ üzerinden paralel kullanmak için parçalara ayırmak, sorguların daha hızlı işlenmesini sağlar. Böyle bir sistemde, birden fazla veri tabanına erişilmesine rağmen, kullanıcı bir tek veri tabanıyla çalışıyormuş gibi işlem yapar .

Dağıtık veri modelleri, veri yoğun uygulamaların gereksinimlerini karşılamak için önemlidir ve genellikle veri büyüklüğü ve işlem hacmi açısından geleneksel veritabanı yönetim sistemlerinin (RDBMS) sınırlarını aşmak için kullanılır. Bu tür modellerin başlıca özellikleri şunlardır:

**Yüksek ölçeklenebilirlik**: Dağıtık sistemler, gerektiğinde daha fazla sunucu veya depolama birimi eklenerek ölçeklenebilir.

**Yüksek kullanılabilirlik**: Veri parçalarının çoğaltılması ve veri işleme işlevlerinin dağıtılması sayesinde, sistemde oluşabilecek tek bir arıza veya kesinti, genellikle tüm sistemi etkilemez.

**Esneklik**: Dağıtık veri modelleri, farklı veri tiplerine ve işlem yüklerine uyum sağlayabilen esnek yapılar sunar.

**Düşük gecikme süreleri**: Verinin fiziksel olarak daha yakın konumlarda bulunması ve paralel işleme yetenekleri, genellikle daha düşük gecikme süreleri sağlar.

**Yerelleştirilmiş veri işleme**: Verinin dağıtıldığı farklı konumlarda işleme yapılabilmesi, yerel veri işleme avantajları sağlar, böylece veri transferi maliyetleri ve gecikme süreleri azaltılabilir.

**Yedeklilik ve güvenlik**: Verinin çoğaltılması ve dağıtılması, veri kaybı riskini azaltır ve güvenliği artırır.

Dağıtık veri modelleri, büyük veri analitiği, bulut bilişimi, akıllı şehirler, IoT (nesnelerin interneti) ve çeşitli diğer veri yoğun uygulama alanlarında yaygın olarak kullanılır. Örnek olarak, Hadoop, Cassandra, MongoDB ve Google Bigtable gibi dağıtık veritabanı sistemleri ve çerçeveleri bu tür uygulamalarda sıklıkla tercih edilir.

VERi TABANI TASARIMI

Veritabanı tasarımı, bilgi yönetimi için önemli bir adımdır. Veritabanı tasarımı, verilerin organize edilmesi, saklanması ve erişilmesi için bir plan oluşturmayı içerir. Başarılı bir veritabanı tasarımı, verilerin etkin bir şekilde depolanmasını, güncellenmesini ve sorgulanmasını sağlar. İşte veritabanı tasarımı adımları:

1. **Gereksinim Analizi**: İhtiyaç duyulan verilerin ve iş süreçlerinin belirlenmesiyle başlar. Hangi verilerin saklanması gerektiği, bu verilerin nasıl kullanılacağı ve iş süreçlerinin nasıl olacağı gibi faktörler değerlendirilir.
2. **Kavramsal Tasarım**: Bu aşamada, gereksinim analizi sonuçlarına dayanarak veritabanının kavramsal bir modeli oluşturulur. Entity-Relationship (Varlık-İlişki) diyagramları gibi araçlar kullanılarak veri yapıları ve ilişkileri tanımlanır.
3. **Mantıksal Tasarım**: Kavramsal tasarımın daha teknik bir düzeyde gerçekleştirildiği aşamadır. Kavramsal model, mantıksal modele dönüştürülür. Tablolar, alanlar ve ilişkiler oluşturulur. Veri türleri, kısıtlamalar ve endeksler tanımlanır.
4. **Fiziksel Tasarım**: Veritabanının fiziksel düzeyde nasıl uygulanacağını belirleyen aşamadır. Bu aşamada, tabloları depolama birimlerine (diskler, bellekler) nasıl yerleştireceğiniz, endeksleme stratejileri ve veri bütünlüğü önlemleri gibi konular ele alınır. Veritabanı yönetim sistemine (DBMS) göre uygun veri depolama ve erişim yöntemleri seçilir.
5. **Uygulama ve Test Etme**: Tasarlanan veritabanının uygulanması ve test edilmesi aşamasıdır. Veritabanı, kullanıcıların ihtiyaçlarını karşılamak ve iş süreçlerini desteklemek için doğru şekilde çalıştığından emin olmak için test edilir. Performans testleri yapılır ve gerektiğinde iyileştirmeler yapılır.
6. **Dağıtım ve Bakım**: Veritabanının kullanıma alınması ve sürekli bakımının sağlanması bu aşamada gerçekleşir. Veritabanı güvenliği, yedekleme ve performans izleme gibi konular düzenli olarak kontrol edilir ve gerektiğinde güncellenir.

Veritabanı tasarımı süreci, verilerin etkin ve güvenli bir şekilde yönetilmesini sağlar ve iş süreçlerinin verimli bir şekilde yürütülmesine katkıda bulunur. Bu adımların dikkatlice izlenmesi, veritabanı tasarımının başarılı bir şekilde tamamlanmasını sağlar.

Formun Üstü



Geleneksel veri tabanı tasarımı, kullanıcı düzeyinden fiziksel düzeye doğrudur. Kavramsal tasarımda, gereksinimlere göre kavramsal şema belirlenir. Kavramsal şema tanımlamada, kavramsal ya da mantıksal veri modelleri kullanılabilir. Kavramsal şema, ortalama veri tabanı kullanıcısı için, veri tabanının yapısını genel olarak tanımlar. Kullanıcıların veri tabanının yapısını anlamalarına ve böylece uygulamalarını modellemelerini sağlar. Kavramsal şema, fiziksel depolama yapılarının ayrıntılarına girmeden, varlıklar, veri tipleri, varlıklar arasındaki ilişki tipleri ve kısıtlayıcılar üzerinde yoğunlaşır. Bu bakımdan kavramsal şema, yüksek düzeyli bir tanımlamadır. Diğer bir ifadeyle, kavramsal şema, yazılım ve donanımdan bağımsızdır ve son kullanıcı tarafından anlaşılması da daha kolaydır.

İlişkisel ve İlişkisel Olmayan Veri Tabanı Sistemleri

İlişkisel ve ilişkisel olmayan veritabanı sistemleri, farklı veri modeli ve organizasyonlarına dayanarak bilgi depolama ve yönetme teknikleridir.

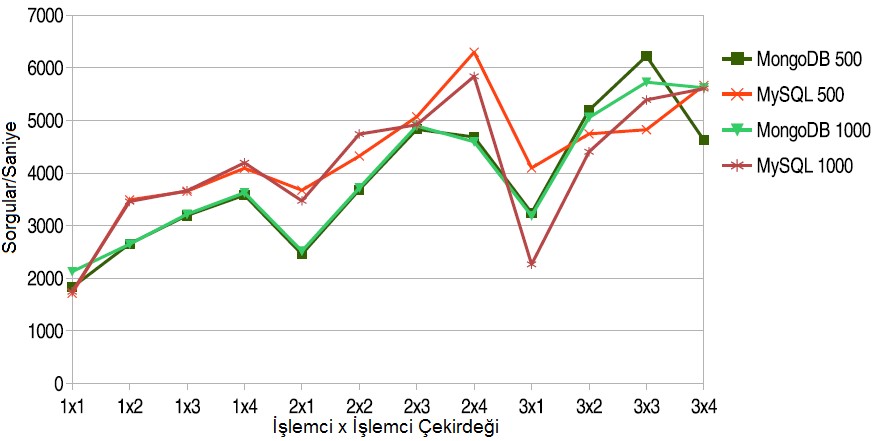
1. **İlişkisel Veritabanı Sistemleri:**
   * İlişkisel veritabanları, verilerin tablo formatında depolandığı ve bu tablolar arasındaki ilişkilerin tanımlandığı sistemlerdir.
   * Veritabanı yönetim sistemleri (DBMS), bu ilişkileri yönetmek için SQL (Structured Query Language) gibi standart sorgu dillerini kullanır.
   * İlişkisel veritabanlarında veriler, tablolar, sütunlar ve satırlar şeklinde düzenlenir. Her tablo bir varlık türünü temsil eder (örneğin, bir müşteri tablosu veya bir sipariş tablosu).
   * Örnek olarak, MySQL, PostgreSQL, Microsoft SQL Server ve Oracle gibi popüler ilişkisel veritabanı sistemleri bulunmaktadır.
2. **İlişkisel Olmayan Veritabanı Sistemleri:**
   * İlişkisel olmayan veritabanı sistemleri, verileri ilişkisel tablolara indirgemek yerine, farklı veri modelleri veya depolama yapıları kullanarak verileri saklar.
   * İlişkisel olmayan veritabanları, özellikle büyük miktarda yapısal olmayan veya yarı yapısal veri depolamak için kullanılır. Bu veri türleri metinler, belgeler, grafikler, ses dosyaları veya video klipler gibi yapısal olmayan veriler olabilir.
   * İlişkisel olmayan veritabanları, ölçeklenebilirlik ve esneklik açısından avantajlar sunabilir. Büyük miktarda veriye anlık erişim sağlama veya dağıtık sistemlerde kullanım gibi senaryolar için idealdir.
   * Örnek olarak, MongoDB (doküman tabanlı), Cassandra (kolon tabanlı), Redis (anahtar-değer tabanlı) ve Neo4j (grafik tabanlı) gibi ilişkisel olmayan veritabanı sistemleri bulunmaktadır.

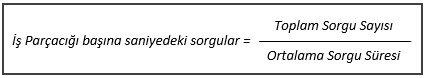
VERİ TABANI MİMARİLERİNİN PERFORMANS KARŞILAŞTIRILMASI

Veritabanı mimarilerinin performansını karşılaştırmak, belirli bir senaryoya, kullanım durumuna ve gereksinimlere göre değişebilir. Ancak genel olarak, ilişkisel ve ilişkisel olmayan veritabanı mimarilerini performans açısından karşılaştırabiliriz.

Bunlar;

* Veri tabanı sunucu sistemleri özellikleri belirlenmesi,
* Veri tabanı şemaları oluşturulması,
* Sorguların belirlenmesi,
* Veri tabanı ayarlarının yapılması,
* Ölçümler ve ölçüm metrikleri bilgileri,
* Performans analizi ve sonuçlarıdır.



Her iş parçacığının saniye saniye sorgu başına nasıl tepki verdiğini ölçmek için aşağıdaki formül kullanılır.

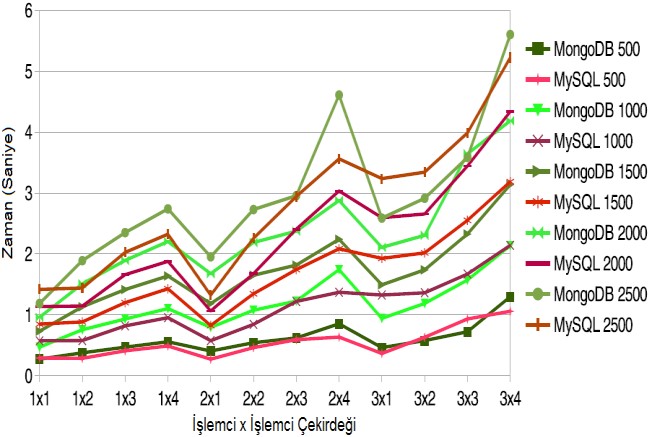
Analiz ve sonuçlar: Burada öncelikle veri tabanlarının farklı sorgu türlerine göre nasıl yanıt verdiği hem okuma hem yazma ile analiz edilen sorguların toplam sayısı ve sonuçları şekiller ile gösterilmiştir.Son olarak veri tabanı boyutunun performansa etkisi konusunda inceleme yapılmıştır. Yapılan açıklanan koşullar kapsamında veri tabanlarının detaylı olarak karşılaştırılabilmesi için çok çeşitli durumlar yaratılmak istenmiştir. Ölçüm için kullanılan yapılandırmalar 1’den 3’e kadar işlemci sayısı ve 1’den 4’e kadar işlemci çekirdek sayısı olarak değişmektedir. Ölçümlerde yapılan sorgu sayısı 500 ile 2500 arasındadır. Her bir ölçüm beş adet test ile bitirilmiştir. Her test sonucunda sorgulardan her birini gerçekleştirmek için alınan ortalama süreler hesaplanarak raporlanmıştır.

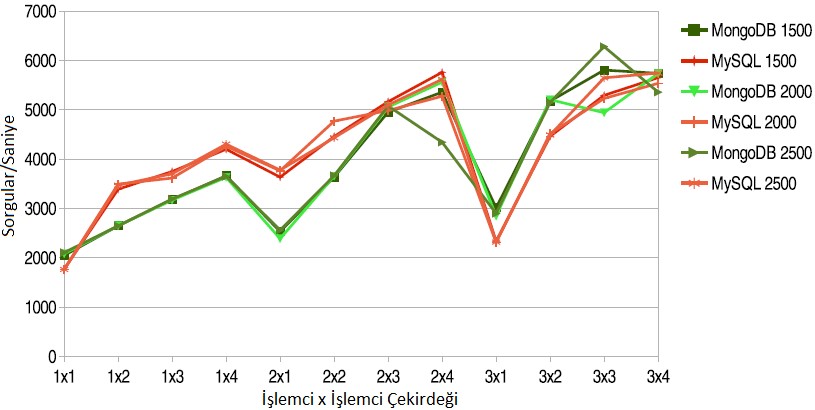
MySQL ve MongoDB veri tabanları sistemlerinin her ikisine eşdeğer miktarda sorgular yapılmaktadır. Bu alandaki yapılan sorgular veri tabanı sorguları bölümünde yer alan sorgu koşullarına göre, ölçümler ise ölçüm metriklerinde tanımlanan formüllere göre hesaplanarak yapılmaktadır.

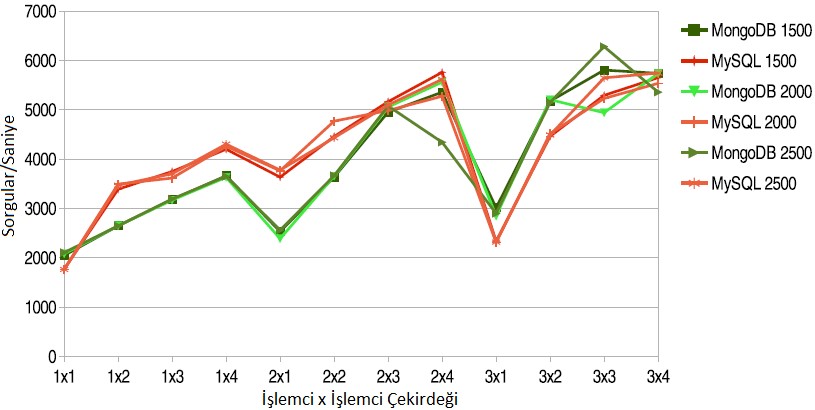
Ölçümler: Projede ölçümler için öncelikle zaman kavramı ön planda tutulması hedeflenmiştir. Zaman ölçümleri için üç yöntem ile hareket edilmiştir.

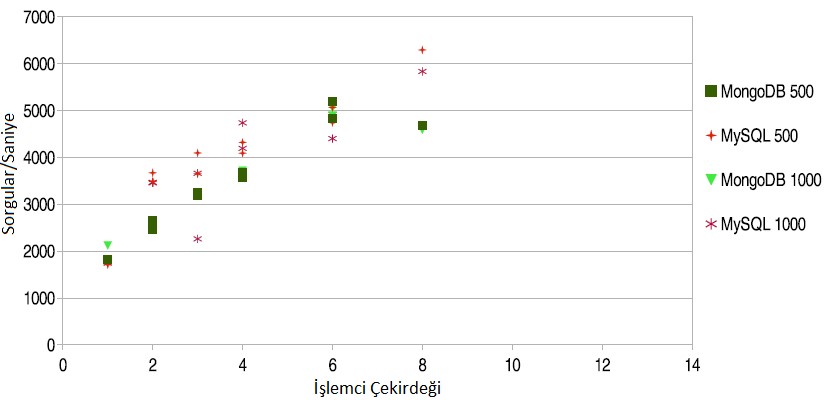
Birinci yöntem; Clock() fonksiyonu kullanımı ile belirli bir süre CPU üzerinde harcanan zaman sonuçlarının elde edilmesini sağlamaktır.

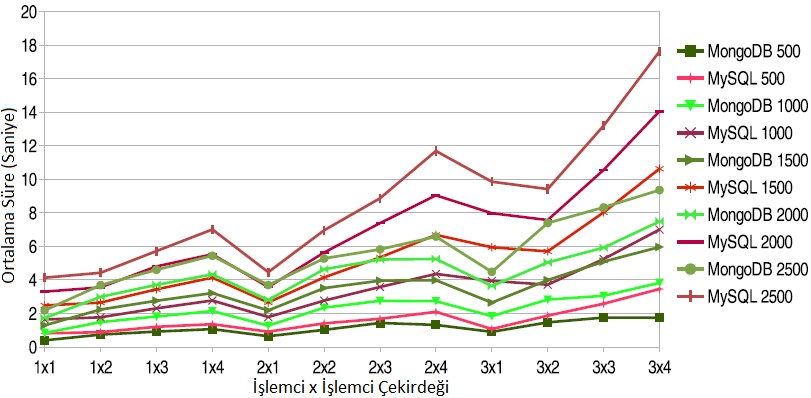
İkinci yöntem; milisaniye hassasiyetiyle zamanlamaları sağlayan Gettimeofday() fonksiyonu kullanılarak sonuçların elde edilmesini sağlamaktır.

 MySQL ve MongoDB veri tabanlarına sorgu 1 (basit sorgu) ile karşılaştırma testi uygulanmıştır. Yapılan analizde; MongoDB, sorgu sayısı farkı arttıkça daha belirgin bir performans kötülüğü gösterdiği tespit edilmiştir. Bu karşılaştırma, işlemci çekirdeği sayılarının toplam sayısı aynı olduğu zaman, 2 ya da 1 işlemci kullanımının değişmez olduğunu açıkça ortaya koymuştur (1x2 ve 2x1). MySQL veri tabanının, özellikle 3 işlemci sayısı ile 1 işlemci çekirdeği sayısına göre incelendiğinde daha kötü performans gösterdiği görülmektedir.

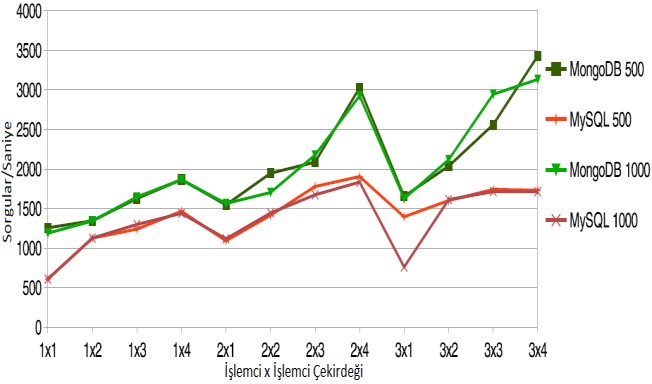
Ayrıca, sorgular/saniye ölçüm metrik grafiği ile de şekil 6.4’de görüldüğü üzere ayrıntılı ortalama süre sonuçları elde edilmiştir.

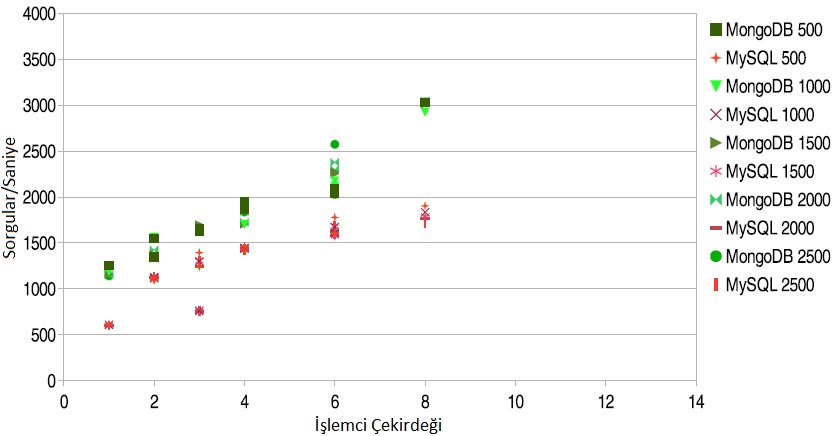
 MySQL veri tabanı sisteminin, sorgu sayıları arttığında MongoDB üzerinde avantaj sahibi olduğu görülmektedir. Fakat 2 işlemci ve 3 işlemci çekirdeği yapılandırmasından sonraki diğer yüksek işlemci-işlemci çekirdeği sayılarında sorgu/saniye grafiğinde keskin bir şekilde azalma görülmektedir. MongoDB bu yapılandırmalarda daha fazla avantaj göstermiştir.

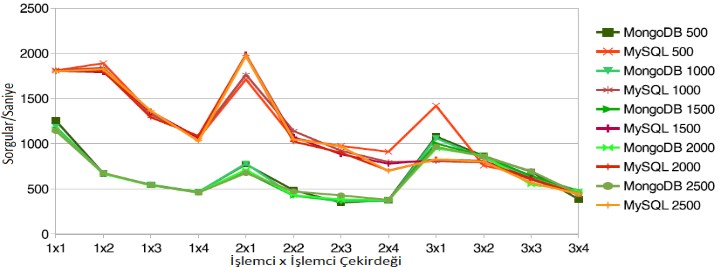
Şekil 6.6’da işlemci çekirdeği miktarı ile saniye başına yapılan sorgu sayıları arasındaki ilişki analizi gösterilmektedir. MySQL için biraz daha iyi olan performans 4 işlemci çekirdeğine kadar hemen hemen aynıdır.

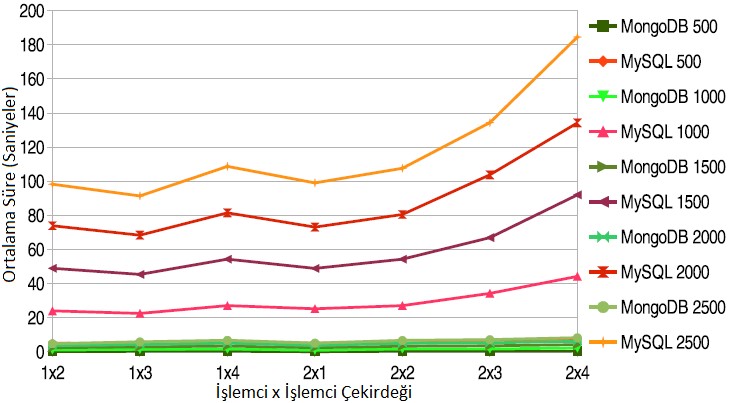
Şekil 6.7’de MySQL ve MongoDB veri tabanlarına ikinci sorgu kodu ile karşılaştırma testi uygulanmıştır. Yapılan analizde; MySQL veri tabanı sisteminin MongoDB’ye göre ortalama sorgu süreleri sonuçları, sorgu sayısı farkı arttıkça daha belirgin bir performans kötülüğü

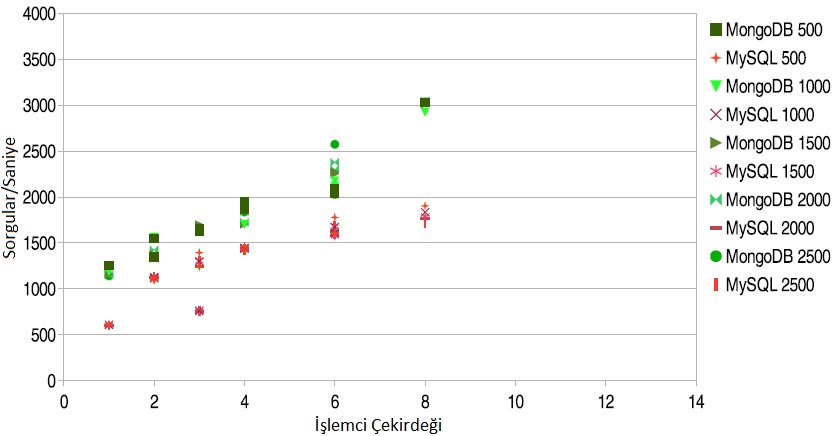
göstermiştir.

Şekil 6.8’de MySQL ve MongoDB veri tabanlarına ikinci sorgu kodu ile karşılaştırma testi uygulanmıştır. Bu test 500 ve 1000 gibi küçük veri kayıtları üzerinde yapılmıştır. Yapılan analizde; MongoDB veri tabanı sisteminin, daha az bir sürede daha çok sorgu yürütmesinin mümkün olduğu, sorgu sayısı değiştikçe performans ölçümünün daha belirgin hale gelerek sorgu/saniye başına %40 oranında daha iyi performans sergilediği gözlemlenmiştir.

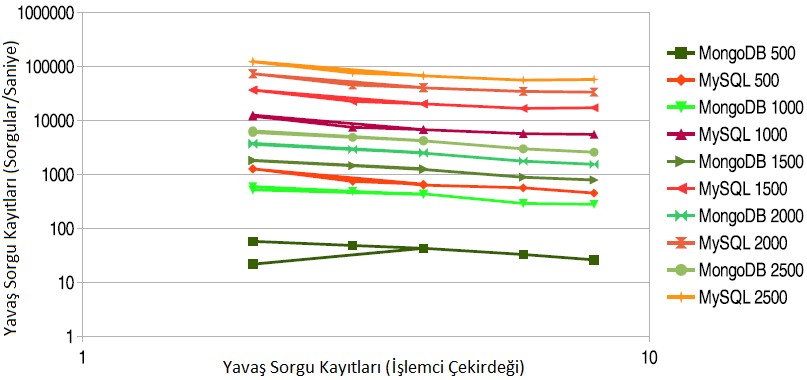
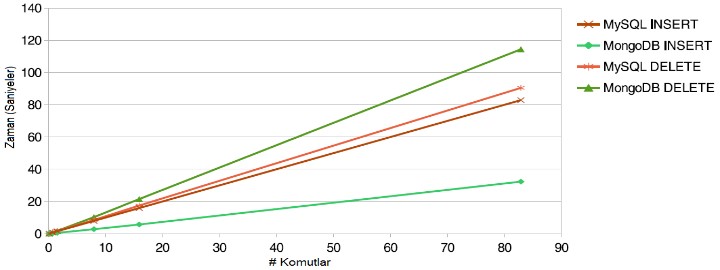
İşlemci çekirdeği miktarı ile saniye başına yapılan sorgu sayıları arasındaki ilişki analizi şekil 6.9’da gösterilmektedir.

Şekil 6.10’da iç içe geçmiş “SELECT” ve “WHERE” işlemlerini içeren üçüncü sorgu neticesinde ortaya çıkan performans değerleri gösterilmektedir.

Şekil 6.12’de üçüncü sorgu ile ortalama süre ölçümleri gösterilmiştir. Yapılan analizde; MySQL veri tabanı sisteminin MongoDB’ye göre ortalama sorgu süreleri sonuçları, veri kayıt sayısı farkı arttıkça oldukça belirgin bir performans kötülüğü gösterdiği gözlemlenmiştir.

MySQL ve MongoDB veri tabanlarına üçüncü sorgu olarak tanımlanan detaylı ve karmaşık sorgu kodu içeren karşılaştırma testi analizi Şekil 6.13’de gösterilmiştir. Bu test 500 ile 2500 veri kayıt setleri üzerinde yapılmıştır. MySQL veri tabanı sistemi, iki eksen boyunca logaritma kullanılarak çizilen grafikte logaritmik bir eğilim olduğu görüntüsü sergilemektedir. MongoDB’nin ise eğilimi nerede ve nasıl gösterdiği net olarak görülmemektedir.

Zamanlama ölçeği büyütülerek veri tabanları sistemleri arasındaki performans farkının Şekil 6.14’de daha anlaşılabilir hale geldiği görülmektedir. Ölçek büyüdüğünde MySQL’in performansındaki dezavantaj açıkça görülmektedir. MongoDB tüm veri kayıt setlerinde oldukça iyi bir performans gösterdiği ortaya konmuştur .

Şekil 6.15’de her iki veri tabanı sisteminin INSERT ve DELETE işlemlerine ait performans grafiği gösterilmektedir. Yapılan analizde; her iki veri tabanının komut sayılarına göre işlem süreleri doğrusal bir eğilim göstermektedir. MongoDB’nin veri ekleme işlemi MySQL’e göre çok daha iyi bir performansa sahiptir. Veri silme işleminde ise MongoDB’nin MySQL ile benzer bir performansa sahip olduğu fakat veri silme komut sayılarının artışı ile MySQL veri tabanı sisteminin silme işleminde iyi bir performans sergilediği gözlemlenmiştir. 

VERİ TABANI MİMARİLERİNİN PERFORMANS KARŞILAŞTIRILMASI ÖZET

Veri tabanı mimarilerinin performansını karşılaştırmak genellikle birçok faktöre bağlıdır ve doğru bir karşılaştırma yapabilmek için spesifik gereksinimlere ve kullanım senaryolarına odaklanmak önemlidir. Ancak, genel olarak, yaygın olarak kullanılan veri tabanı mimarilerini karşılaştırabiliriz:

1. İlişkisel Veri Tabanı Yönetim Sistemleri (RDBMS): İlişkisel veri tabanları, geleneksel olarak yapılandırılmış veriler için kullanılır ve SQL dilini kullanarak veriye erişim sağlar. Örnekler arasında MySQL, PostgreSQL ve Oracle gibi sistemler bulunur. RDBMS'ler genellikle yüksek veri bütünlüğü ve ACID (Atomicity, Consistency, Isolation, Durability) uyumluluğu sağlar.
2. NoSQL Veri Tabanı Yönetim Sistemleri: NoSQL veri tabanları, ilişkisel olmayan veri modellerine ve esnek şemalara odaklanır. Örneğin, belge tabanlı veritabanları (MongoDB), anahtar-değer tabanlı veritabanları (Redis), sütun tabanlı veritabanları (Cassandra) gibi farklı türlerde NoSQL veritabanları bulunur. NoSQL sistemleri genellikle büyük ölçekli dağıtılmış sistemlerde performans avantajları sunar.
3. NewSQL Veri Tabanı Yönetim Sistemleri: NewSQL, ilişkisel veritabanlarının ölçeklenebilirlik ve performans sorunlarını ele almak için tasarlanmıştır. Geleneksel RDBMS'nin ACID özelliklerini koruyarak, dağıtılmış sistemlerde ölçeklenebilirlik sağlar. NewSQL örnekleri arasında VoltDB ve Google Spanner bulunmaktadır.
4. Graf Veri Tabanları: Graf veritabanları, graf yapıları üzerinde veri saklamak ve sorgulamak için optimize edilmiştir. Örnekler arasında Neo4j ve Amazon Neptune bulunur. Graf veritabanları, ilişkisel olmayan veri modelleri ile ağ yapısı veya sosyal medya analizi gibi senaryolarda performans avantajı sağlar.
5. Bellek İçi Veri Tabanları: Bellek içi veritabanları, verileri bellekte saklayarak yüksek performans sağlarlar. Örnekler arasında Redis ve Memcached bulunur. Bu tür veritabanları, özellikle hızlı erişim gerektiren uygulamalarda kullanılır.

Performans karşılaştırması yaparken, belirli kullanım senaryolarına göre ölçümler yapılmalıdır. Performansı etkileyen faktörler arasında veri hacmi, sorgu türleri, eşzamanlı kullanıcı sayısı, yük dengesi, yedekleme ve geri yükleme süreleri gibi birçok faktör bulunmaktadır. Ayrıca, ölçeklenebilirlik, yüksek kullanılabilirlik, güvenlik ve maliyet gibi diğer faktörler de dikkate alınmalıdır.

Formun Üstü

SONUÇ VE DEĞERLENDİRME :Yapılan analizlerde NoSQL ağırlıklı bir veri tabanının büyük miktarda veri çiftleri içerebildiği, veri çoğaltmada da basit şeması nedeniyle MongoDB kullanılarak daha hızlı daha karmaşık sorgu tiplerinin çalıştırılabildiği izlenmiştir. Her iki veri tabanı sisteminde farklı yapılandırma durumlarında ikinci sorgu tipi ile yapılan performans testlerinde, MongoDB veri tabanı sistemi MySQL’e göre en iyi performansı göstermiştir. Sonraki karşılaştırma testlerinde detaylı ve karmaşık sorgular ele alınarak MongoDB, alt belge koleksiyonu kullanımı nedeniyle MySQL üzerinde çok büyük bir avantaja sahip olduğunu göstermiştir. Bu avantaj, veri tekrarı yaşanabilme durumu pahasına dikkate değer bir şekilde görülmüştür. Bu tür sorgularda büyük veri tabanı boyutundan kaynaklanan depolama ve bellek miktarı maliyetini hesaplarken alternatif olarak NoSQL veri tabanlarını dikkate almak çok önemlidir. Yapılan son performans testleri ise yazma ve silme işlemleridir. Basit arama sorgularında mantıksal olarak veri silme işlemi dikkate alınarak yapılan karşılaştırmalar sonucunda MySQL veri tabanı sistemi iyi bir performans göstermiştir. Öncelikle silinecek verinin bulunması gerektiğinden silme işlemine direk olarak bağlantılanır. Her iki veri tabanı bu karşılaştırma sonucunda doğrusal bir eğilim gösterirken MongoDB eklemeler sırasında MySQL’e göre oldukça belirgin ve çok daha iyi bir performans göstermiştir. Yapılan testlerin bir diğer önemli yönü, işlemci ve işlemci çekirdeklerinin farklı şekillerde yapılandırılarak kullanılmasıdır. Bu çalışmada, veri tabanlarının 1, 2 ve 3 işlemci ile birden fazla işlemci çekirdeği üzerinde nasıl performans göstereceğinin test edilmesi ve karşılaştırılması için çoklu sorgulama işlemleri kullanılmıştır. Bu test detaylı ve karmaşık sorgular ile birlikte yapılmıştır. Burada sorgu karmaşıklığına bağlı olarak veri tabanları farklı davranmaktadır. Bu farklılık ise daha fazla sorgu sayısı ile sorgular/saniye analiz grafiklerinde görülmüştür. Çok çeşitli durumlar yaratılarak elde edilen sonuçlar ile işletmelere hangi durumda nasıl bir veri tabanı yönetim sistemi kullanmaları konusunda fikir verilmiştir. Sonuç olarak, farklı kriterler ile bu veri tabanlarını incelediğimizde iki veri tabanının da avantaj ve dezavantajları olduğu görülmüştür.