T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

# YÜKSEK LİSANS TEZİ

# İLİŞKİSEL VE İLİŞKİSEL OLMAYAN VERİTABANI YÖNETİM SİSTEMLERİNİN KARŞILAŞTIRILMASI VE PERFORMANS ANALİZİ

**ÖMER COŞKUN**

**DANIŞMAN**
**Prof. Dr. AHMET SERTBAŞ**

**II. DANIŞMAN**
**Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Bilgisayar Mühendisliği Programı**

**İSTANBUL-2019**

Bu çalışma 31.05.2019 Tarihinde aşağıdaki jüri tarafından
Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı Yüksek Lisans Tezi olarak kabul edilmiştir.

TEZ JÜRİSİ

Prof. Dr. Ahmet SERTBAŞ
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Yusuf YASLAN
İstanbul Teknik Üniversitesi
Bilgisayar ve Bilişim Fakültesi

Dr.Öğr.Üyesi Muhammed Ali AYDIN
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

# ÖNSÖZ

Yüksek Lisans tez çalışmam süresince, bana yol gösteren bilgi ve desteğini esirgemeyen değerli hocam ve tez danışmanım Prof. Dr. Ahmet SERTBAŞ'a,

Çalışmalarım süresince yardımlarıyla beni sürekli destekleyen, bilgi ve birikimlerini benimle paylaşan Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN'a

Ayrıca bana desteklerini hiç bir zaman esirgemeyen, sabırları ve destekleriyle tüm öğretim hayatım boyunca her zaman yanımda olan çok sevdiğim aileme,

Eşime,

Sonsuz teşekkürler...

Mayıs 2019 ÖMER COŞKUN

# İÇİNDEKİLER

**Sayfa No**

# ŞEKİL LİSTESİ

**Sayfa No**

# TABLO LİSTESİ

**Sayfa No**

# SİMGE VE KISALTMA LİSTESİ

| Kısaltmalar | Açıklama |
|---|---|
| **ACID** | : Bölünmezlik, tutarlılık, izolasyon, dayanıklılık (Atomicy, concistancy, isolation, durability) |
| **API** | : Uygulama programlama arayüzü (Application programming interface) |
| **BASE** | : Kolay ulaşılabilirlik, esnek durum, nihai tutarlılık (Basic availability, soft state, eventual consistency) |
| **CPU** | : Merkezi İşlem Birimi (Central processing unit) |
| **ER Model** | : Varlık-ilişki modeli (Entity-relationship model) |
| **JSON** | : Javascript nesne gösterimi (Javascript object notation) |
| **OS** | : İşletim sistemi (Operating System) |
| **SQL** | : Yapısal sorgulama dili (Structured query language) |
| **UML** | : Birleşik modelleme dili (Unified Modelling Language) |
| **VTYS** | : Veri tabanı yönetim sistesmi |

# ÖZET

## YÜKSEK LİSANS TEZİ

### İLİŞKİSEL VE İLİŞKİSEL OLMAYAN VERİTABANI YÖNETİM SİSTEMLERİNİN KARŞILAŞTIRILMASI VE PERFORMANS ANALİZİ

**Ömer COŞKUN**

**İstanbul Üniversitesi-Cerrahpaşa**

**Lisansüstü Eğitim Enstitüsü**

**Mühendislik Bilimleri Anabilim Dalı**

**Danışman : Prof. Dr. Ahmet SERTBAŞ**

**II. Danışman : Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN**

İlişkisel olmayan veri tabanı yönetim sistemleri yakın geçmişte ortaya çıkmış ve gün geçtikçe yaygınlaşıp kullanım oranını oldukça arttırmıştır. Bu tez kapsamında ilişkisel ve ilişkisel olmayan veri tabanlarının eşit koşullarda veri ekleme, veri silme, veri sorgulama gibi temel işlevlerle birlikte karmaşık sorgulardaki performans analizi yapılmıştır. İlişkisel olmayan veri tabanı yönetim sistemleri, yazılım geliştirme açısından esnekliği, yatay ölçeklenebilirliği ve yüksek performansıyla ön plana çıkmıştır. Yapılan performans analizi sonucunda organizasyonların maliyet ve performans açısından kendileri için en uygun veri tabanı yönetim sistemini seçebilmesi amaçlanmıştır.

Mayıs 2019, 53 sayfa.

**Anahtar kelimeler:** İlişkisel veri tabanı yönetim sistemleri, ilişkisel olmayan veri tabanı yönetim sistemleri (NoSQL), MongoDB, MYSQL

# SUMMARY

## M.Sc. THESIS

### COMPARISON OF RELATIONAL AND NON-RELATIONAL DATABASE MANAGEMENT SYSTEMS AND PERFORMANCE ANALYSIS

**Ömer COŞKUN**

**Istanbul University-Cerrahpasa**

**Institute of Graduate Studies**

**Department of Engineering Sciences**

**Supervisor : Prof. Dr. Ahmet SERTBAŞ**

**Co-Supervisor : Assoc. Prof. Dr. Gülsüm Zeynep GÜRKAŞ AYDIN**

Non-relational database management systems have emerged in the recent past and have become widespread and have increased the usage rate considerably. Within the scope of this thesis, performance analysis of complex questions is carried out together with basic functions such as data insertion, data deletion, data querying, and relational and non-relational databases. Non-relational database management systems have come to the forefront in terms of software development, flexibility, horizontal scalability and high performance. As a result of performance analysis, it is aimed that organizations can choose the most suitable database management system for themselves in terms of cost and performance.

May 2019, 53 pages.

**Keywords:** Relational database management systems, non-relational database management systems (NoSQL), MongoDB, MYSQL

# 1. GİRİŞ

Günümüzde teknolojinin hızla gelişmesine paralel olarak ortaya çok büyük miktarda veri ortaya çıkmaktadır. Özellikle son yıllarda akıllı telefon ve tablet kullanımındaki artış, sosyal medya ve elektronik ticaretin yaygınlaşması internet ortamında üretilen verilerin çok daha büyük boyutlara ulaşmasını sağlamıştır.

Veri tabanı yönetim sistemlerinde en sık kullanılan model ilişkisel veri tabanı yönetim sistemidir. Veri tutarlılığından ödün vermeyen bankalar gibi kuruluşlar için ilişkisel veri tabanı yönetim sistemleri hala vazgeçilmezdir. Fakat internet ortamında üretilen ve genellikle yapısal olmayan veriler yaygın olarak kullanılan ilişkisel veri tabanı yönetim sistemlerinde saklanmaya uygun değildir. Bu sebeple ilişkisel veri tabanı yönetim sistemlerinin yetersiz kaldığı yerlerde ortaya ilişkisel olmayan veri tabanı yönetim sistemleri çıkmıştır.

Son yıllarda ortaya çıkan büyük verileri işlemek için ilişkisel olmayan veri tabanı yönetim sistemlerine yönelmek kaçınılmazdır. Ölçeklenebilme maliyetinin düşük olması ve yüksek veri okuma, yazma hızına sahip olması nedeniyle büyük verileri işleyen Google, Facebook, Amazon gibi şirketler ilişkisel olmayan veri tabanlarını tercih etmişlerdir.

Bu tez çalışması kapsamında, ilişkisel ve ilişkisel olmayan veri tabanları ile ilgili literatürdeki gelişmeler incelenmiştir. Literatürde, ilişkisel ve ilişkisel olmayan veri tabanlarının avantaj ve dezavantajların inceleyen, ilişkisel olmayan veri tabanlarını kendi içerisinde inceleyen ve karşılaştırma yapan çalışmalar gözlemlenmiştir. İlişkisel olmayan veri tabanlarının son yıllardaki kullanım miktarının ciddi şekilde artması ile ilişkisel ve ilişkisel olmayan veri tabanlarının organizasyonlar için seçim problemi ortaya çıkmıştır. Bu problem özelinde yapılan birkaç çalışma dışında yeterli çalışmaların yapılmadığı görülmüştür. Özellikle yapılan çalışmalarda ilişkisel ve ilişkisel olmayan veri tabanlarını karşılaştırırken, karşılaştırma yapılan sorgu seçimleri ve test edilen veri boyutunun yetersiz miktarda olduğu tespit edilmiştir. Bu çalışmanın literatüre katkısı açısından incelenen bazı çalışmalar şunlardır;

Diogo ve diğ. [1], çalışmalarında NoSQL veri tabanlarının tutarlılık modellerini incelemişlerdir. İlişkisel veri tabanlarına göre tutarlılık problemleri halen devam eden ilişkisel

olmayan veri tabanlarından en popüler beş veri tabanını seçerek tutarlılıklarını karşılaştırmışlardır.

Khan ve diğ. [2], çalışmalarında ilişkisel ve grafik tabanlı NoSQL veri tabanlarını karşılaştırarak tahmini performans analizi yapmışlardır. Karşılaştırma kapsamında Oracle ve Neo4j veri tabanlarında aynı sorgular çalıştırılarak performans analizleri ortaya konmuştur.

Nguyen [3], yazdığı tez kapsamında e-ticaret ve sosyal medya uygulamaları üzerinde MongoDB ve VoltDB veri tabanları uygulamalı olarak karşılaştırmıştır.

Al-Saeedi [4], yüksek lisans tezinde B2C web uygulamalarında uygun NoSQL veri tabanı seçimi üzerinde durmuştur.

Niyizamwiyitira ve Lundberg [5], yayımlanan çalışmalarında bir kümeleme de ilişkisel ve ilişkisel olmayan veri tabanlarını karşılaştırmışlardır. Çalışma kapsamında, farklı boyutlardaki üç veri kümesinde yazma, okuma, gecikme ve en yakın komşu sorgularında veri tabanlarının verimlilikleri tespit edilmiştir.

Abromova ve diğ. [6], yayımlanan makalelerinde ilişkisel olmayan veri tabanları içinde performansın veri tabanı türü ile ilişkisini değerlendirmişlerdir.

Öztürk ve Atmaca [7], yayımladıkları makalede ilişkisel ve ilişkisel olmayan veri tabanlarının performansını yönetim bilişim sistemleri kapsamında incelemişlerdir. Veri tabanı modelinin seçilmesi, performans ölçümleri yapılması gibi konulara değinmişlerdir.

Gökşen ve Aşan [8], çalışmalarında ilişkisel ve ilişkisel olmayan veri tabanlarını inceleyerek mimari yapıları hakkında bilgi vermiş ve bir takım temel sorgularda performans ölçümleri yapmışlardır.

Kumar ve diğ. [9], yayımlanan çalışmalarında MYSQL ve MongoDB veri tabanlarını üzerinde temel ekleme, silme ve seçim sorgularını kullanarak bir takım performans ölçümleri yapmışlarıdır.

Dumanlı[10], yüksek lisans tezinde en çok kullanılan NoSQL veri tabanlarını performans yönünden karşılaştırmıştır.

Shwaysh[11], yüksek lisans tezinde, NoSQL veri tabanlarını kendi içerisinde güvenlik ve performans yönünden mukayese etmiştir.

Sethi ve diğ. [12], yayımlanan çalışmalarında NoSQL veri tabanlarının özelliklerini ve veri tabanı modellerinin her birinin gücünü ve sınırlamalarını ele almıştır.

Bu tez çalışmasında ilişkisel ve ilişkisel olmayan veri tabanı yönetim sistemleri incelenecek ve her bir yönetim sistemi için seçilen iki veri tabanı üzerinde yüksek sayıda veri ile birlikte karmaşık sorgulama işlemleri yapılarak karşılıklı performans analizi yapılacaktır. Tezin ikinci bölümünde, kapsamlı bir literatür çalışması sonucunda veri tabanı ve veri tabanı yönetim sistemleri kavramsal olarak incelenecektir. Üçüncü bölümde, tez çalışması kapsamında performans analizinin yapılabilmesi için gerekli yöntemler ve materyaller belirlenecektir. Dördüncü bölümde, belirlenen yöntem ve materyaller kullanılacak ve performans analizleri yapılacaktır. Ayrıca bu analiz sonucunda elde edilecek bulgular saptanacaktır. Son bölümde ise performans analizi sonucu ortaya çıkan bulgular değerlendirilecek ve veri tabanı yönetim sistemi seçimi hakkında tespitler yapılacaktır.

# 2. GENEL KISIMLAR

İçinde bulunduğumuz teknoloji çağında bilgi hayatımızın çok önemli bir parçası haline gelmiştir. Bilgisayarlar, cep telefonları, tabletler gibi cihazlar ve web sayfaları, mobil uygulamalar, sosyal medya gibi araçlar aracılığı ile hayatımızın her anında sürekli bir veri akışı gerçekleşmektedir. Bu veri akışı bilgiye ulaşmak için büyüm bir önem arz etmektedir. Çünkü bilgi, verinin işlenmemiş halidir.

Veri kavramının TDK'ya göre sözlük anlamı "gerçek"tir. Veri, bilginin ham halidir. Veri, tek başına kullanılamaz ve herhangi bir anlam ifade etmez. Veri, analiz edilmeye, gruplandırılmaya, yorumlanmaya ihtiyaç duyar. Veriler bilgi hiyerarşisinin en alt basamağadır. Sayısal, sözel ya da bir sembol olabilirler. Anlamlandırılarak, analiz edilerek, sınıflandırılarak bilgi haline dönüştürülürler. Sensörlerden, gözlemlerden, araştırmalardan vb. birçok alandan elde edilebilir [13].

Veriler şu şekilde gruplandırılabilir [14].

• Yapılandırılmış, Yapılandırılmamış, Yarı yapılandırılmış

• Statik, dinamik, akan

• Güvenli / açık, özel / halka açık

• Ücretli / ücretsiz

• Açık hükümet verisi

• Açık veri

• Büyük veri

Veri tabanı genel tanımı itibariyle, kullanım amaçlarına uygun olarak bir araya getirilmiş ve düzenlenmiş veri topluluklarıdır. Birbiri ile ilişkisi olan verilerin beraberce saklandığı, fiziksel ve mantıksal tanımlara sahip olan bilgi depolarıdır. Veri tabanı, ilişkili nesnelere ve bu nesnelerin ilişkilerini modeller [15].

Veri tabanı, bankalar, hastaneler, üniversiteler, şirketler gibi çeşitli organizasyonların operasyonel verilerinin tamamını saklamaktadır. Bu kuruluşların çalışabilmesi, yönetilebilmesi ve gündelik işlerine devam edebilmesi için kullanılan çeşitli veriler veri tabanları aracılığıyla saklanmaktadır.

Veri tabanları ile saklanan tüm verilere sistematik olarak erişim imkânı bulunmaktadır. Ayrıca saklanan veriler yönetilebilir, güncellenebilir, taşınabilir ve birbirleri arasında ilişki kurulabilir.

Veri tabanlarını veri tekrarını önler ya da en az seviyeye indirir. Veri ve bilgi güvenliğini ve güvenirliğini sağlar. Veri bütünlüğünün bozulmasını engeller.

## 2.1. VERİ TABANI

### 2.1.1. Veri tabanı Tarihçesi

İnsanlar çok uzun zaman önce bilgi depolamaya başladılar. Antik çağda, devlet daireleri, kütüphaneler, hastaneler ve iş organizasyonları tarafından ayrıntılı veri tabanı sistemleri geliştirildi ve bu sistemlerin bazı temel prensipleri bugün hala kullanılıyor.

1960'lı yıllarda bilgisayar kullanımının özel kuruluşlar için de uygun maliyetli bir seçenek haline gelmesiyle birlikte veri tabanı kullanımı yaygınlaşmaya başladı. Bu yıllarda iki popüler veri modeli vardı. COADSYL adlı bir ağ modeli ve IMS adlı bir hiyerarşik model. Bu yıllarda veri tabanlarına yapılan ilk büyük yatırım IBM'in American Airlines havayollarının rezervasyon verilerini saklamak için SABRE isimli sistemi kullanmasıdır [15].

1970 yılında Edgar Frank Codd tarafından önerilen makale insanların veri tabanı konusundaki düşüncelerini tamamen değiştirdi. Önerdiği ilişkisel veri tabanı modelinde veri tabanı şemasının veya mantıksal organizasyonun fiziksel bilgi depolama birimiyle ilişkisi kesildi ve bu ilke veri tabanları için standart hale gelmiştir [16].

Günümüzde veri tabanları her yerdedir ve günlük yaşamın akışı içinde her alanda kullanılmaktadır. Web sayfaları, mobil uygulamalar, üniversiteler, hastaneler vb. birçok uygulama ve kuruluş veri tabanları ile birlikte çalışmaktadır.

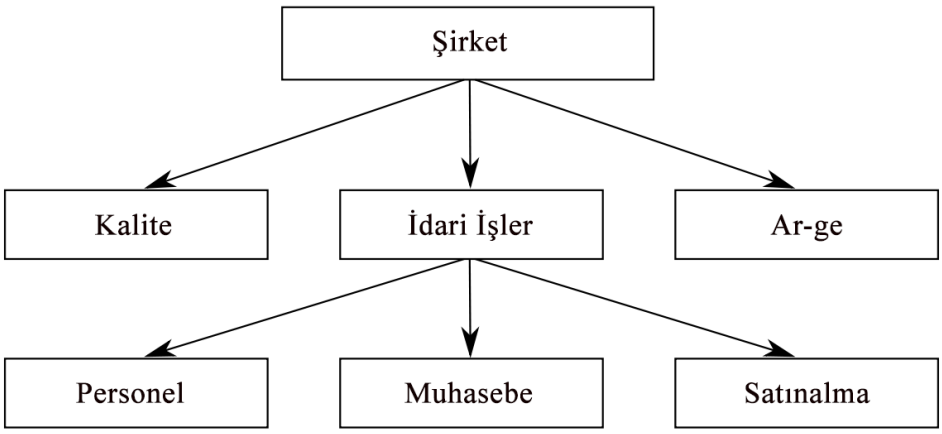
### 2.1.2. Veri tabanı Modelleri

Geçmişten günümüze kadar birçok veri modeli geliştirilmiştir. Veri modeli, kavramsal olarak verinin veri tabanı yönetim sistemleri üzerinde ne şekilde gösterileceğini temsil eden modeldir. Veri modeli, hangi verinin ne şekilde sınıflandırılacağına ve gerekliliğine odaklanmaktadır. Diğer bir deyişle verinin mimari planını göstermektedir.

Veri modellerini temel olarak 4 grupta toplamak mümkündür [17].

#### *2.1.2.1. Hiyerarşik Veri Modeli*

İlk olarak kullanılan veri modelidir.1960 ve 1970'li yıllarda kullanılmıştır. Adını veriyi saklama yönteminden dolayı almıştır. Bu modelde veri tabanının sakladığı verilere kayıt adı verilmiştir. Kayıtlar ağaç(tree) yapısında saklanır. İlk kayıt kök ismini alır. Kök kaydın bir veya daha fazla çocuk kayıtları olabilir. Her çocuk kayıt kendi içinde çocuk kayıtlara sahip olabilir. Kök kayıt dışındaki bütün kayıtlar mutlaka bir ebeveyn kayda sahiptir. Çocuk kayıtlar sadece tek bir noktadan ebeveyn kayda bağlanabilir [18].

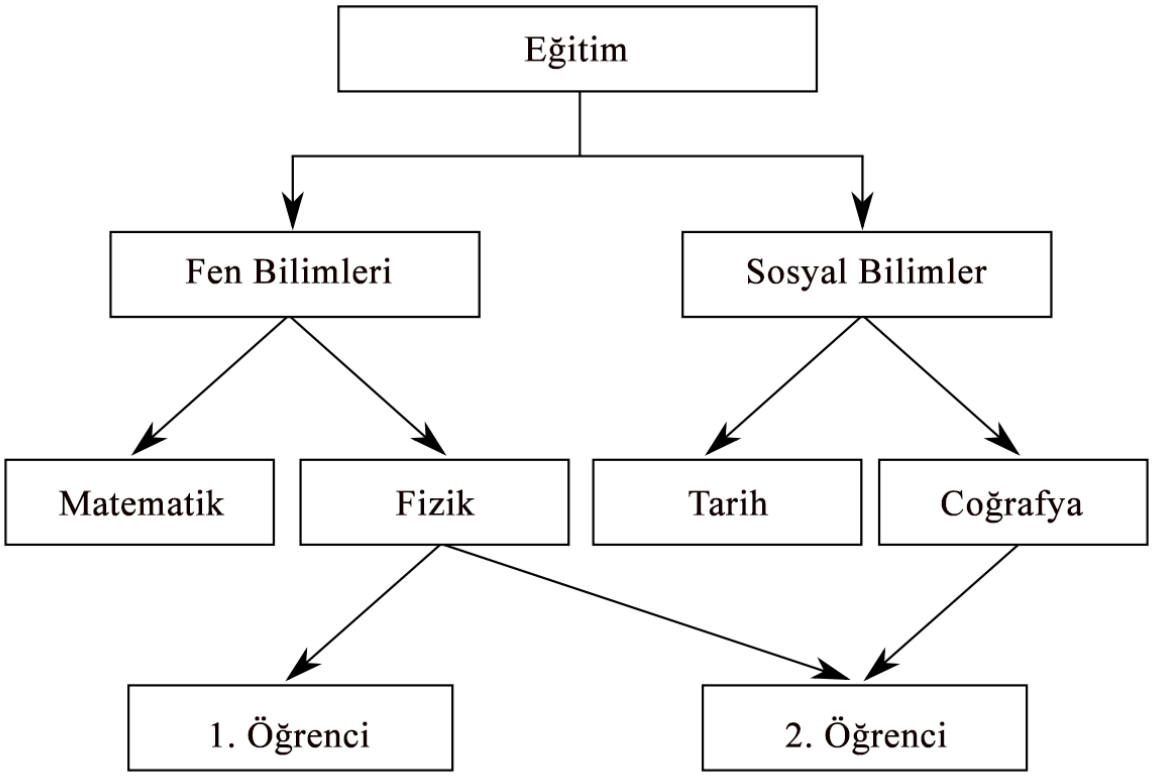



**Şekil 2.1:** Hiyerarşik veri modeli örneği.

#### *2.1.2.2. Ağ Veri Modeli*

1970'li yıllarda ve 1980'li yılların ilk yarısında kullanılmıştır. Hiyerarşik modelin yetersiz olması nedeniyle kullanılmaya başlanmıştır. Hiyerarşik modelin geliştirilmiş hali de denebilir. Hiyerarşik modelden farklı olarak bir veri birden fazla veri ile ilişki kurabilmektedir. Verilerin

ağ yapısında bağlandığı varsayılmaktadır. Ebeveyn – çocuk ilişkisi hiyerarşik modelden farklıdır. Her kayıt birden fazla ebeveyn ve çocuk kayda sahip olabilir. Böylelikle model bire bir ve bire çok ilişkilere destek vermektedir. Veri tekrarını büyük ölçüde azaltır [19].

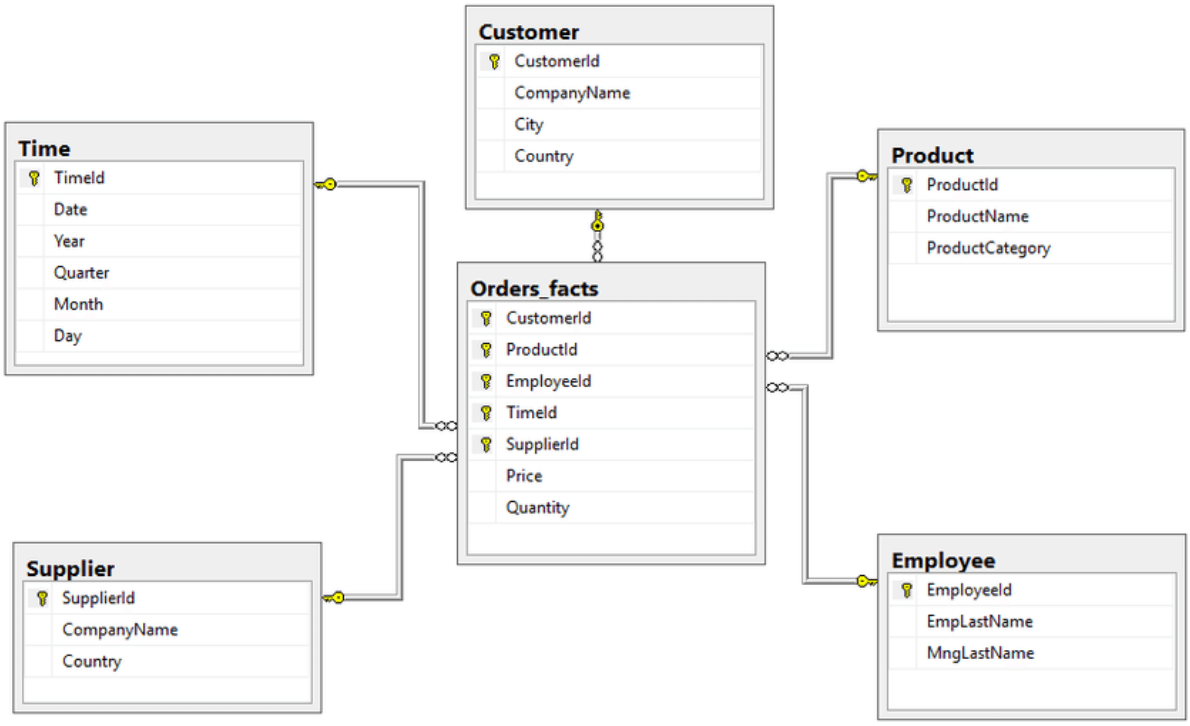



**Şekil 2.2:** Ağ veri modeli örneği.

### *2.1.2.3. İlişkisel Veri Modeli*

Bu model 1985'li yıllarda yaygınlaşarak 1990'lı yıllarda hemen hemen tüm sistemlerde kullanılmaya başlanmıştır. Hiyerarşik model ve ağ modelinin yetersiz kalması sonucu ortaya çıkmıştır. Günümüzdeki veri tabanlarının çoğu ilişkisel modele destek vermektedir. Modelin temeli ilişkiye dayanmaktadır. Bu modelde veriler satır ve sütunlardan oluşan tablolar yardımıyla depolanmaktadır. Her sütun alan (field) adı verilen ayrı bir veriyi saklamaktadır. Her alanın saklayacağı veri tipi önceden belirlenmiştir. Aynı satırlarda saklanan veriler ise aynı kaydı ifade etmektedir. Tablolar üzerinde ilişkiler ve veriler saklanmaktadır. İlişkiler birincil

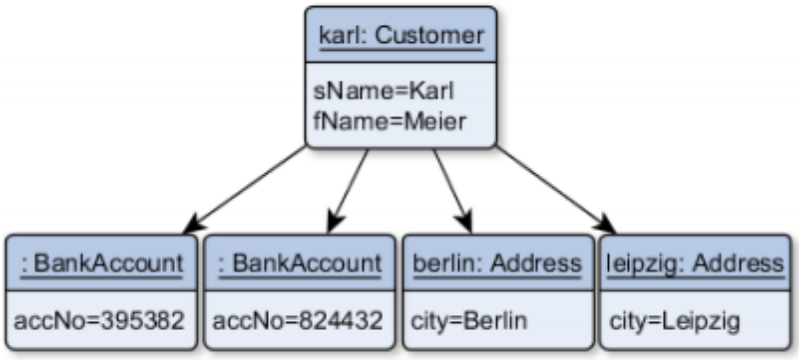
anahtar (primary key) ve ikincil anahtar (foreign key) ile kurulur. Anahtarlar sayesinde indeksleme yapıldığı için ilişkisel model performans açısından diğer modellerden çok daha iyi durumdadır. Tablolar arasında kullanılan ortak alanlar ilişkilendirildiği için raporlama konusunda da çok esnektir [18].




**Şekil 2.3:** İlişkisel veri modeli.

### *2.1.2.4. Nesneye Yönelik Veri Modeli*

Son yıllarda ortaya çıkan bu model giderek yaygınlaşmaktadır. İlişkisel veri modeli ile birlikte kullanılmaktadır. Bu modelde bir sorguya ait mutlaka daha önce tanımlanmış nesne kümesi olması gerekiyor. Belirli özelliklerdeki nesneyi bir kez tanımladıktan sonra aynı özelliklere sahip işlemlerde tekrar tekrar kullanmayı sağlamaktadır. Nesne tabanlı veri tabanlarında arama işlemleri çok hızlı bir şekilde yapılmaktadır. Özellikle çok büyük verilere sahip tablolar üzerinde arama yaparken ilişkisel modele göre çok daha performanslı çalışmaktadır. İlişkisel veri modelinde veriler iki boyutlu tablo olarak gelirken nesneye yönelik veri modelinde tek parça halinde gelmektedir. Bu da birden fazla veri dönen durumlarda nesneye yönelik veri modelinde performans sorunları ortaya çıkarmaktadır. İlişkisel veri modelinden, nesneye yönelik veri modeline geçiş maliyeti çok yüksek olduğu için bu model ticari anlamda başarısız olmuştur [20].




**Şekil 2.4:** Nesneye yönelik veri modeli örneği [21].

### 2.1.3. Veri tabanı Tasarımı

İhtiyaca yönelik olarak ilgili konuyu oluşturan verilerin, veri tabanı üzerinde nasıl tutulacağının modellenmesine veri tabanı tasarımı denir. Bir veri tabanı tasarımı temel olarak dört aşamada gerçekleştirilir. Bu aşamaların aynı sırada ilerlemeyebilir. Hatta her aşamada bir önceki aşamaya dönme ve değişiklik yapma ihtiyacı duyulabilir [22].

#### *2.1.3.1. Gereksinimlerin Belirlenmesi*

Kullanıcının gereksinimleri belirlenir. Veri tabanı oluşturulacak konu ile ilgili veriler toplanır. Toplanan bilgiler çerçevesinde veri tabanında kullanılacak veri tipleri, veri yapıları ve ver grupları belirlenir.

#### *2.1.3.2. Kavramsal Model*

Kavramsal model bir nevi hazırlık aşaması olarak değerlendirilebilir. Bu seviyede model oluşturulurken, kullanıcı gereksinimlerini ifade eden, geniş çaplı bir model ortaya koymak gerekir. Bu model, saha analizinden gelen gereksinimleri karşılayacak nitelikte olmalıdır. Bu aşamada veri tabanı tasarlarken kullanılabilecek birçok yöntem bulunmaktadır. ORM ile modelleme, UML ile modelleme, ER ile modelleme en sık kullanılan yöntemlerdir [22].

#### *2.1.3.3. Mantıksal Model*

Mantıksal model, veri tabanı seçimi yapılarak kavramsal modelin veri tabanı üzerinde uygulanmasıdır. Kavramsal modelin aksine veri tabanına, veri modeline donanım ve

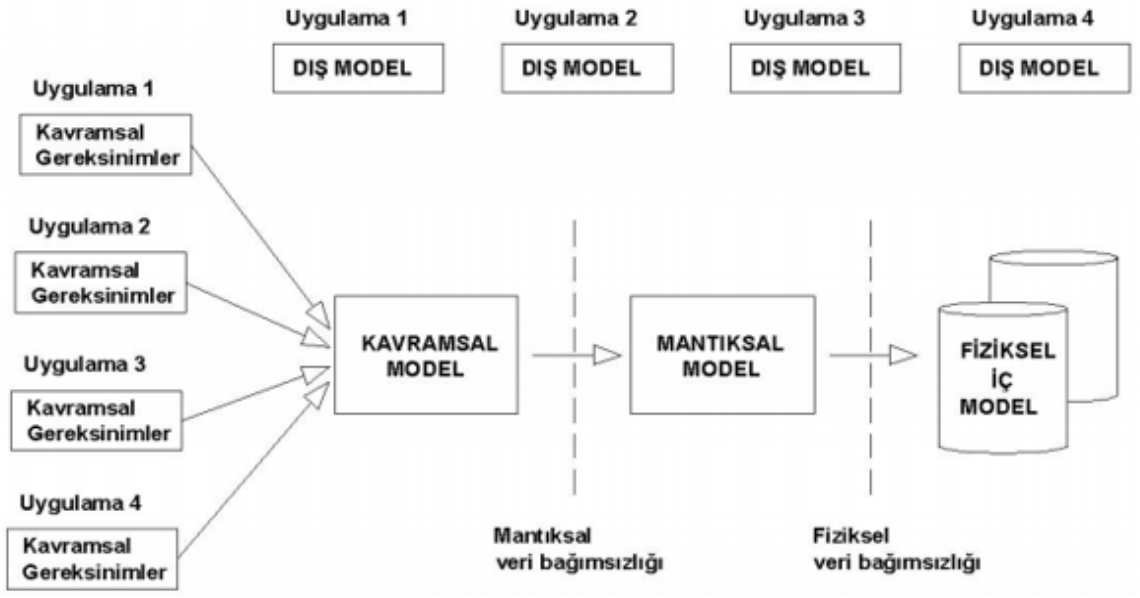
yazılım birimlerine bağımlılıkları mevcuttur. Mantıksal model uygulanmadan önce, kavramsal modele en uygun veri modeli seçilmelidir. Varlık - ilişki modelinden mantıksal modele geçiş yapılır [22].

### *2.1.3.4. Fiziksel Model*

Fiziksel model veri tabanının fiziksel olarak oluşturulması için, uygun tekniklerin belirlenmesi ve gerçekleştirilmesidir. Fiziksel model tasarlandıktan sonra kurgulanarak ihtiyaçlara uygunluğu ve performansı denetlenmelidir. Bu modelde asıl amaç minimum maliyetle maksimum performans elde etmektir [22].

Fiziksel model temel olarak dört basamakta uygulanır.

- Veri gösterimi şeklinin belirlenmesi,
- Erişim metotlarının seçilmesi,
- Kümeleme,
- Veri tabanının yüklenmesi




**Şekil 2.5:** Veri tabanı fiziksel tasarım modeli [23].

### 2.1.4. Veri tabanı Sorgulama Dilleri

Veri tabanı kullanımı ile birlikte depolanan verileri sorgulama ihtiyacı ortaya çıkmıştır. Veri tabanı sorgulama dilleri, veri tabanları üzerinde karmaşık sorgular yapmak için tasarlanmış dillerdir.

İlk olarak matematiksel sözdizimine sahip olan SQUARE adlı bir dil geliştirilmiştir. Daha sonra geniş kitlelerin daha kolay kullanabilmesi için İngilizce 'ye benzer bir söz dizimine sahip olan SEQUEL dili geliştirilmiştir. Zamanla bu dil SQL olarak adlandırılmıştır ve günümüzde birçok veri tabanı tarafında kullanılmaktadır. SQL dışında PL-SQL, T-SQL, GraphQL, LINQ, AQL, vb. birçok sorgulama dili geliştirilmiştir [24].

### 2.1.5. Veri tabanı Performansı

Veri tabanlarının temel amacı bilgiyi depolamak ve gerektiğinde en hızlı şekilde o bilgiye erişmektir. Bu yüzden bir veri tabanından en büyük beklenti performanslı çalışmasıdır. Veri tabanı bakım işlemleri ve performans iyileştirmeleri veri tabanı yöneticilerinin sorumluğundadır.

Veri tabanlarının performanslı çalışabilmesi birkaç seviyede performans optimizasyonları yapılmalıdır.

#### *2.1.5.1. Veri tabanı performans optimizasyonu*

Veri tabanlarında veriye kesintisiz erişim olmalıdır, bu nedenle veri tabanında oluşabilecek sorunlar büyümeden çözülmelidir. Veri büyüme hızına dikkat edilmelidir. Veri tabanı için yeterli fiziksel kaynak ayrılmalıdır. Veri tabanı üzerinde performans kaybı yaşatan sorgular takip edilmeli ve gerekli iyileştirmeler yapılmalıdır [13].

#### *2.1.5.2. Uygulama performans optimizasyonu*

Veri tabanını kullanan uygulamaların veri tabanı ile etkileşimine dikkat edilmelidir. Veri tabanı ile iletişim kuran sorgular optimize edilmeli, veri tabanında performans sorunu yaşatacak sorgulardan kaçınılmalıdır. Sorgular gerekirse uygulama üzerinden veri tabanı üzerine taşınmalıdır [13].

#### *2.1.5.3. Sistem performans optimizasyonu*

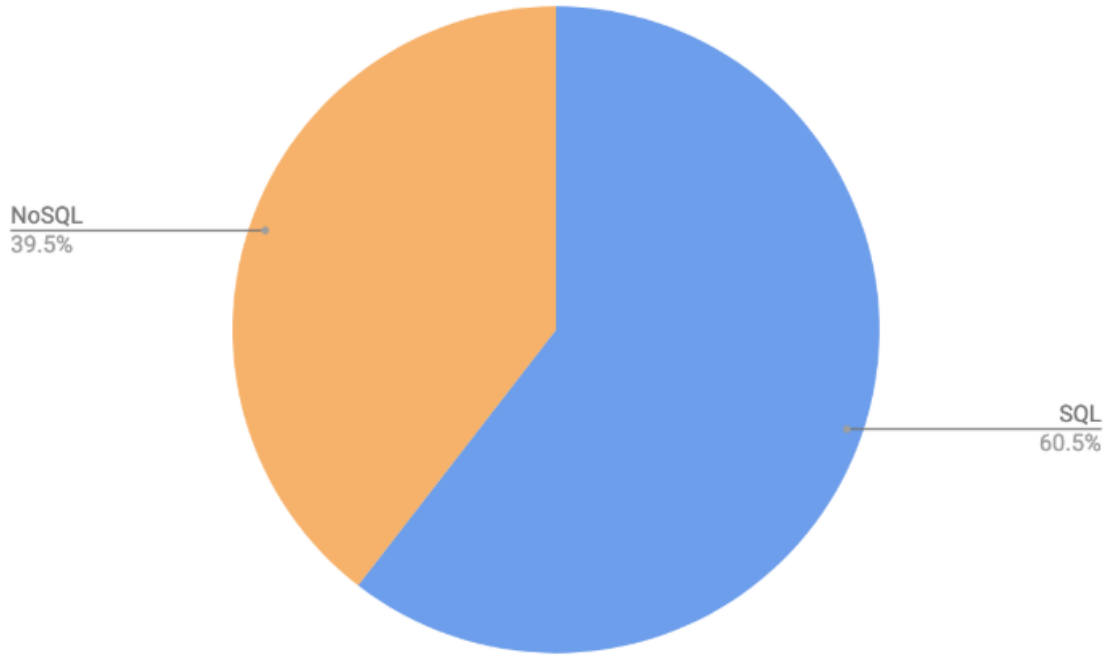
Bu seviyedeki performans iyileştirmeleri sistem yöneticileri tarafından yapılmalıdır. Veri tabanı yöneticilerinin, sistem yöneticilerine yapacağı önerilerle seviyesinde performans iyileştirmeleri yapılır. Bu kapsamda veri tabanın üzerinde çalışacağı sistem için gerekli ve yeterli kaynak sağlanmalıdır [13].

#### *2.1.5.4. Ağ performans optimizasyonu*

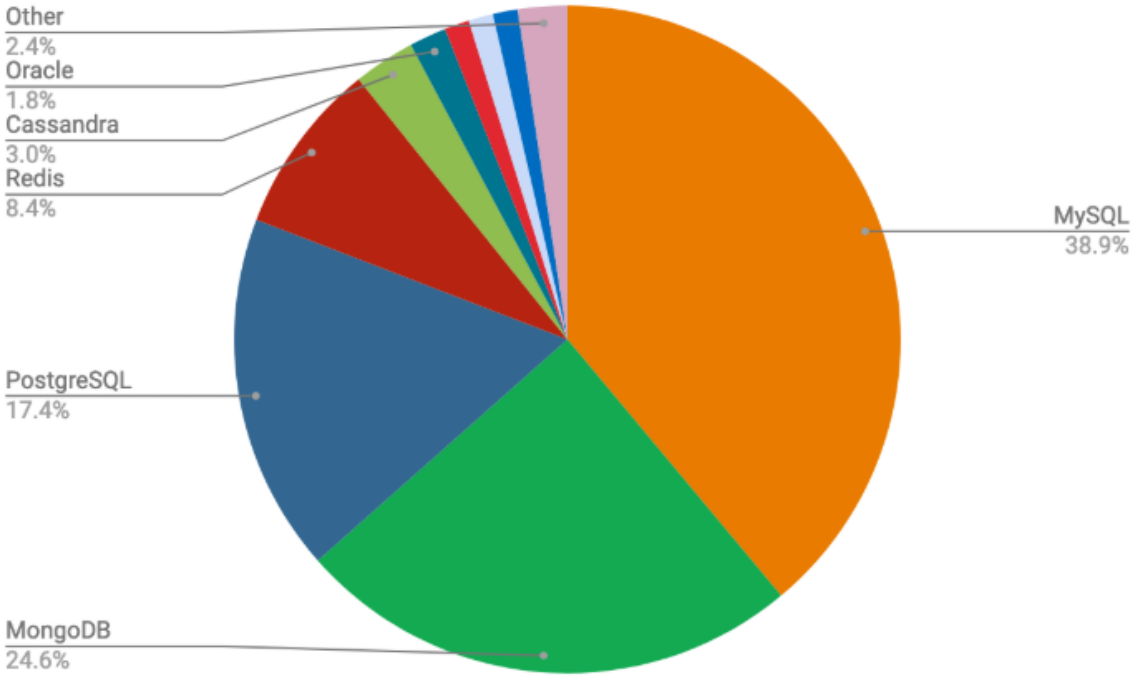
Veriye erişimin olduğu ağ ortamı kullanıcıların düzgün çalışabilmesi için yeterli olmalıdır. Veri tabanı yöneticisi uygulama ve sistemi değerlendirirken aynı zaman ağ performansını da göz önünde bulundurmalıdır [13].

### 2.1.6. Sektöre Sık Kullanılan Veri tabanları

Günümüzde 300'den fazla ilişkisel ve ilişkisel olmayan veri tabanı bulunmamaktadır. Çoğunlukla ilişkisel veri tabanları kullanılsa da her geçen gün ilişkisel olmayan veri tabanlarının popülaritesi daha da artmaktadır.




**Şekil 2.6:** İlişkisel ve ilişkisel olmayan veri tabanlarının pazardaki dağılımı [25].




**Şekil 2.7:** Sektörde sık kullanılan veri tabanlarının pazardaki dağılımı [25].

## 2.2.VERİ TABANI YÖNETİM SİSTEMLERİ

Veri tabanları oluşturmak, düzenlemek, geliştirmek ve bakımını yapmak gibi birçok işlemin bir arada yapılabildiği yazılımlara Veri tabanı Yönetim Sistemi denir. Kısaca VTYS olarak adlandırılır.

Veri tabanı yönetim sistemleri, verinin fiziksel olarak saklanmasını, erişilebilmesini ve sorgulanabilmesini sağlayan kurallar sistemidir. Aynı anda birden çok bağlantıyla veriye erişim imkânı vermektedir. Veriye erişimde kullanıcıları rol ve yetkiler ile kısıtlayabilmektedir. Günümüzde MSSQL, Oracle, MySQL, PostgreSQL gibi birçok ticari ya da açık kaynak kodlu VTYS'ler bulunmaktadır.

Veri tabanı yönetim sistemlerini kullanmanın faydaları şöyle sıralanabilir.

**Veri tutarlılığı (Data Consistency):** Veri tek bir yerde saklandığı için herkes aynı veriye erişmektedir ve böylece veri tutarlılığı sağlanmış olur [26].

**Veri Tekrarı (Data Redundancy):** Verinin sağlanmasında merkezi bir yapı olduğu için verinin tek yerden yönetilmesini sağlayarak verinin tekrarını engeller [26].

**Veriye Eşzamanlı Erişim (Data Concurrency):** Birden çok kullanıcının aynı anda aynı veriye tutarlı bir şekilde eşzamanlı olarak erişebilmesini sağlar [26].

**Verinin bütünlüğü (Data Integrity):** Veriye erişimi tutarlı ve doğru bir şekilde sağlar. Silinen bir veri, ilişkili olduğu tüm yerlerden silinerek veriye erişim bir bütün olarak sağlanmış olur [26].

**Veri Güvenliği (Data Security):** Veriye sadece erişim izni olan kullanıcıların erişmesini sağlayarak veri güvenliğini sağlar. Aynı zamanda verinin istenmedik bir şekilde bozulmasını, değişime uğramasını engeller [26].

**Veri Bağımsızlığı (Data Independence):** Veri tabanında saklanan verinin yapısını ve yönetimini kullanıcıdan soyutlar, kullanıcı bu işlemlerle ilgilenmek zorunda kalmaz [26].

### 2.2.1. İlişkisel Veri Tabanı Yönetim Sistemleri

Günümüzde en fazla kullanılan veri tabanı yönetim sistemidir. Satır ve sütunların birleşiminden meydana gelen tablolardan oluşur. Bu tabloların birbirleriyle ilişkileri mevcuttur. Her bir tablo, kendisi üzerinde tanımlanan verileri uygun veri yapısında tutmak üzere tasarlanmıştır. Ayrıca bu sistem üzerinden saklanan tablolarda kayıtlar eşsiz anahtar değere sahip olmalıdır [27].

1970’ yıllarda kullanıma başlanan ilişkisel veri tabanı yönetim sistemleri günümüzü kadar birçok gelişme kaydetmiştir. Bu sistemlerin üzerinde geliştirilmiş veri tabanı uygulamalarının çoğunda indeksleme, fonksiyon oluşturma, prosedür oluşturma, performans yönetimi, yedekleme ve güvenlik sağlanması gibi bir çok işlem yapılabilmektedir [27].

Bu yönetim modelinde veri bağımsızdır. Bu sayede kayıtların fiziksel olarak yerleşimi ve veri yapısı kullanıcı için önem arz etmez. Tablo içindeki bir sütun birincil anahtar olarak belirlendiği için her bir satır benzersiz hale gelir. Birincil anahtar değeri ile kullanıcı tablo içinde aradığı veriye erişebilir. Günlük hayatımızda kullandığımız TC kimlik numaramız birincil anahtara doğal bir örnektir [27].

### 2.2.2. İlişkisel Olmayan Veri Tabanı Yönetim Sistemleri

İlişkisel olmayan veri tabanları Carlo Strozzi tarafından 1998 yılında tasarlanmıştır. SQL dilini kullanmayan ve birbirleriyle ilişki içermeyen verilerden oluşan bir sistemdir. Bu sebeple NOSQL olarak da adlandırılır. NOSQL modelinin ortaya çıkmasına, internet üzerinde oluşan ve yapısal olmayan büyük miktarlardaki veriler sebep olmuştur. Bu verilerin ilişkisel veri tabanı yönetim sistemleri ile yönetilmesi performans sorunları ortaya çıkarmaktadır. Bu noktada ilişkisel olmayan veri tabanı modellerinden çok yüksek performanslar elde edilmektedir [28].

**Esneklik:** NOSQL esnek veri modeli ve esnek şema yapısı nedeniyle yüksek hızda yazılım geliştirmeyi mümkün kılar.

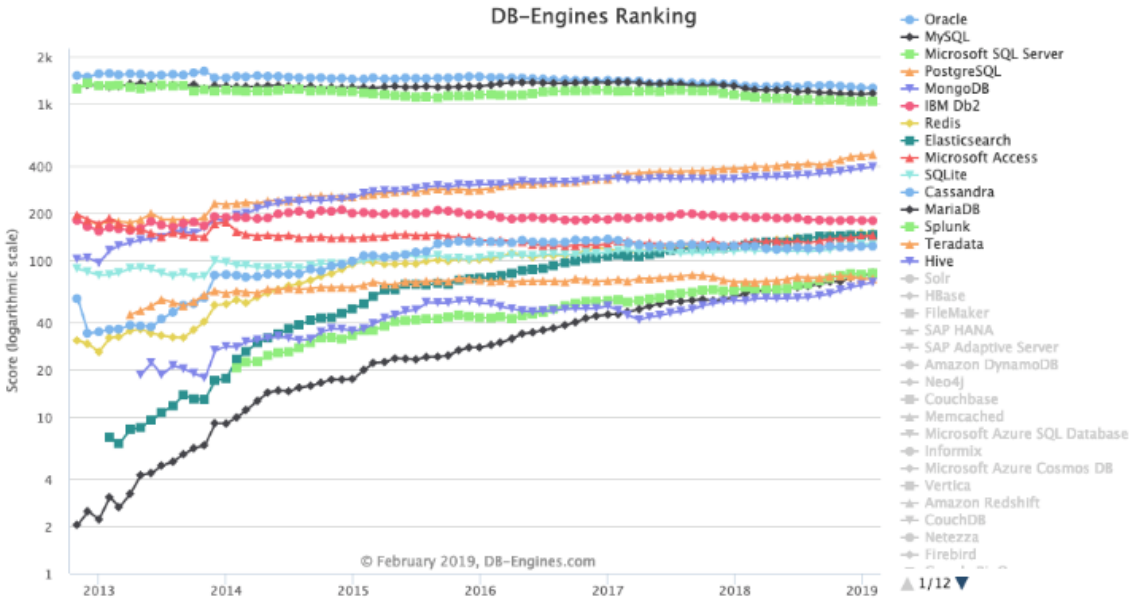
**Ölçeklenebilirlik:** NOSQL veri tabanları yüksek maliyetli sunuculara ihtiyaç duymaz. Dağıtım donanım yapıları kullanılarak genişleyebilecek şekilde tasarlanmıştır.

**Yüksek Performans:** NOSQL veri tabanları ilişkisel veri tabanları ile kıyaslandığında çok daha yüksek performans gösterecek yapıda geliştirilmiştir.

**İşlevsellik:** NOSQL veri tabanları, işlevsel API'ler ve veri türleri sağlayacak yapıdadır.

İlişkisel olmayan veri tabanlar gün geçtikçe daha popüler hale gelmektedir. Günümüzde veri tabanı pazarının yaklaşık %40'ını ilişkisel olmayan veri tabanları oluşturmaktadır [25].

Şekil 2.8'de veri tabanı yönetim sisteminden bağımsız popüler veri tabanlarının kullanım miktarların yıl yıl gösterilmektedir. Sektörde en çok kullanılan NoSQL veri tabanı olan Mongo DB'nin kullanım oranı son 5-6 yıllık süreçte yaklaşık olarak 4 kat artış göstermiştir.




**Şekil 2.8:** Veri tabanı uygulamalarının yıllara göre popülerliği [25].

#### *2.2.2.1. Doküman tabanlı veri tabanları*

Bu tür NOSQL veri tabanlarında bir anahtar değere karşılık olarak genellikle JSON formatında dokümanlar saklanır. Dokümanlar birçok alan barındırabilir ve her bir dokümanın yapısı birbirinden farklı olabilir. En önemli özellikleri esnek olmalarıdır. Bu yüzden esnek veri yapısına ihtiyaç duyan uygulamalarda rahatlıkla kullanılabilir. MongoDB, Couchbase, Firebase en çok kullanılan doküman tabanlı veri tabanlarıdır [29].

#### *2.2.2.2. Anahtar - Değer tabanlı veri tabanları*

Çok yüksek sayıda ve küçük boyutta okuma yazma işlemlerinin yapıldığı uygulamalarda bu tür veri tabanları kullanılır. Bir anahtara değere karşılık küçük boyutlu basit veriler tutulur. Önbellek yazılımlarında sıklıkla kullanılır. Redis, Memcahed, Hazelcast en çok kullanılan anahtar-değer tabanlı veri tabanlarıdır [29].

#### *2.2.2.3. Grafik tabanlı veri tabanları*

Veriler düğüm, ilişki ve özellik şeklinde tutulurlar. Diğer veri tabanlarına göre farkı veriyle birlikte ilişkileri de tutmasıdır. Dağıtık olarak çok sayıda sunucu da çalışabilir. Büyük boyutta

ki verileri saklayabilir. Kısa süreli veri de tutarsızlık sergileyebilir. Neo4j, FlockDB, JanusGraph en çok kullanılan grafik tabanlı veri tabanlarıdır [29].

## 2.3.İLİŞKİSEL VE İLİŞKİSEL OLMAYAN VERİ TABANI YÖNETİM SİSTEMLERİNİN KARŞILAŞTIRILMASI

### 2.3.1. Veri yapıları

İlişkisel veri tabanlarında veriler, tablonun veri yapısına bağımlı şekilde saklanmaktadır. Bu veri yapılarının temel amacı tablolara girilen verilerin tutarlılığını sağlamaktır. İlişkisel bir veri tabanında nesneleri saklamaya yarayan tablo ve şemalar mevcuttur bunların tamamına "Meta-Data" adı verilmektedir. İlişkisel olmayan veri tabanlarında ise veriler yapısal olmayan şekilde ve ya yarı yapısal olarak saklanmaktadır. İnternet ortamında yapısal olmayan verinin hızla artışı bu tür veri tabanlarının önemini ortaya çıkarmaktadır [30].

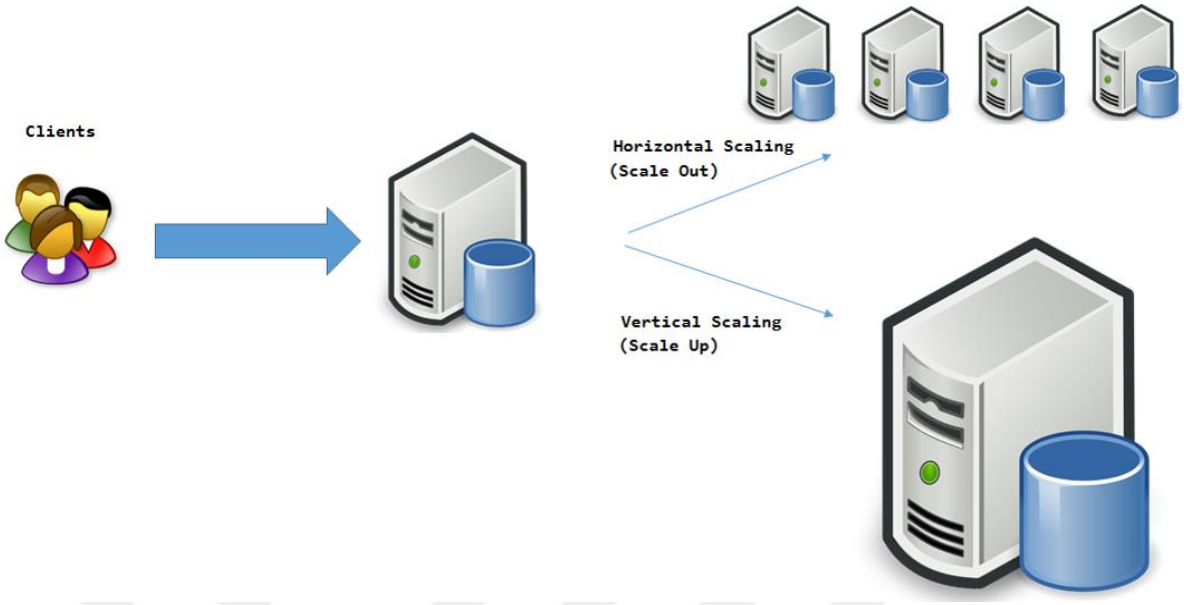
İki sistemin veri yapılarını karşılaştırdığımızda ilişkisel veri tabanı yönetim sistemi tutarlılık konusunda daha avantajlı iken ilişkisel olmayan veri tabanları esneklik konusunda daha avantajlıdır.

### 2.3.2. Ölçeklendirme

İlişkisel veri tabanı yönetim sistemleri dikey ölçeklenebilir, ilişkisel olmayan veri tabanı yönetim sistemleri yatay ölçeklenebilir veri tabanlarıdır

Dikey ölçeklenebilir veri tabanlarında, veri sayısı arttıkça buna paralel olarak sunucu disk, ram ve işlemci gücünü arttırmak gerekmektedir. Dikey ölçekleme de tek sunucu olması sunucu yönetimi açısından avantajdır. Fakat sunucu donanım özelliklerinin artması ciddi maliyetleri de beraberinde getirecektir. Ayrıca tek sunucu olması, sunucuda oluşacak sorunlarda tüm sistemi etkilemektedir [31].

Yatak ölçeklenebilir veri tabanlarındaki çözüm sunucu sayısını arttırmaktır. Mevcut sunucu ya da sunucularda donanım gücü iyileştirmesi yapmadan yeni sunucular eklenerek sistem performansı arttırılır. Dikey büyümeye oranla çok daha ucuz maliyetlidir. Bir sunucuda oluşacak sorun da sistem kesintiye uğramaz ve çalışmaya devam eder [31].




**Şekil 2.9:** Veri tabanı yönetim sistemlerinde yata ve dikey ölçeklendirme [32].

### 2.3.3. ACID ve BASE kuralları

İlişkisel ve ilişkisel olmayan veri tabanları farklı temel prensiplere göre çalışırlar. İlişkisel veri tabanları ACID ile ifade edilen prensiplere göre çalışırken, ilişkisel olmayan veri tabanları BASE ile ifade edilen prensiplere göre çalışır. ACID prensipleri; "bölünmezlik, tutarlılık, yalıtım, dayanıklılık" kelimelerinin İngilizce kısaltmalarından oluşmaktadır [33].

**Atomicity (Bölünmezlik):** Başlayan bir işlemin tamamen tamamlanmasını ya da hiç tamamlanmamasını gerektirir. Yarım kalan bir işlem verilerde tutarsızlığa neden olabilir bu yüzden başlatılan bir işlem bütün olarak değerlendirilir [33].

**Consistency (Tutarlılık):** Tamamlanan bir işlem sonucunda girdi ve çıktıların şema kurallarına uygunluğunu ifade eder [33].

**Isolation (İzolasyon):** Bir işlem başlatıldığında etki ettiği alana bu işlem dışında müdahale edilemeyeceğini ifade eder [33].

**Durability (Dayanıklılık):** Eğer bir işlem başarılı şekilde tamamlandıysa sistem bunu güvence altına alabilmelidir [33].

BASE prensipleri; “kolay ulaşılabilirlik, esnek durum ve nihai tutarlılık” kelimelerinin İngilizce kısaltmalarından oluşmaktadır.

**Basic availability (Kolay ulaşılabilirlik):** Sistem her gelen talebe cevap verir. Fakat hata durumunda bile cevap verme zorunluluğundan dolayı veri tutarlılığı garanti edilmez ve verinin tamamına erişim sağlanamayabilir [34].

**Soft state (Esnek durum):** Anlık veri tutarlılığı sağlanmadığı için verinin tüm cihazlarda aynı olma zorunluluğu böylece ortadan kalkmış olur [34].

**Eventual consistency (Nihai tutarlılık):** Verilerde yapılan güncellemeler tüm cihazlara belirli bir süre sonunda yansıyabilir. Böylece anlık veri tutarlılığını garanti edemese de nihai olarak veri tutarlılığını garanti etmiş olur [34].

### 2.3.4. Performans

İlişkisel veri tabanı yönetim sistemlerinde performans genellikle disk sistemine bağlıdır. Performansı maksimum seviyeye çıkarabilmek için sorgular, indeksler ve tablo yapıları optimize edilmelidir.

İlişkisel olmayan veri tabanı yönetim sistemlerinde performans ağ gecikmelerine, istek yapan uygulamaya ve kullanılan donanımın küme boyutuna bağımlıdır.

# 3. MALZEME VE YÖNTEM

## 3.1. VERİ TABANLARI SEÇİMİ

Çalışmaya başlamadan önce genel bir araştırma yapılarak, kurumların temel veri tabanı yönetim sistemi ihtiyacını çözebilecek düzeyde en uygun ilişkisel ve ilişkisel olmayan veri tabanı yönetim sisteminin seçilmesi hedeflenmiştir.

Kıyaslanacak veri tabanı sistemleri için, ilişkisel veri tabanı yönetim sistemi olarak MySQL 5.7.3 veri tabanı, ilişkisel olmayan veri tabanı yönetim sistemi olarak ise MongoDB 3.6 veri tabanı kullanılmıştır.

İlişkisel veri tabanı yönetim sistemi için MySQL 5.7.3 sisteminin seçilmesinde;

- Ölçeklenebilirlik ve Esneklik
- Yüksek Başarım
- Yüksek Erişilebilirlik
- Web ve Veri Ambarlarında Güçlü
- Güçlü Veri Koruma
- Kapsamlı Uygulama Geliştirme
- Kolay Yönetilebilir
- Açık Kaynak Kod Özgürlüğü
- Düşük Maliyet

faktörleri etkili olmuştur [10].

İlişkisel olmayan veri tabanı yönetim sistemi için MongoDB 3.6 sisteminin seçilmesinde;

- Gelişmiş Güvenlik
- Kolay Yönetim
- Esnek Veri Modeli
- Ölçeklenebilir
- Sürdürülebilir
- Hızlı

faktörleri etkili olmuştur [35].

## 3.2. KULLANILAN VERİ TABANI SUNUCUSU SİSTEM ÖZELLİKLERİ

Bu çalışmada, kullanılan sunucuya ait teknik özellikler Tablo 3.1'de yer almaktadır.

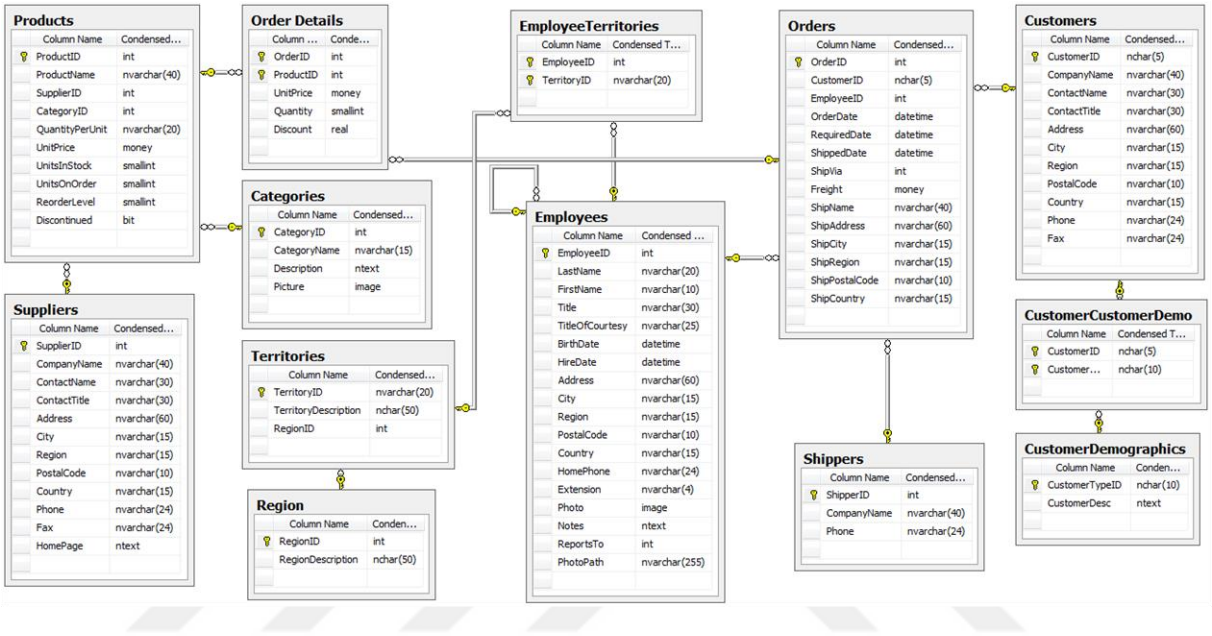
**Tablo 3.1:** Fiziksel ve sanal sunucu sistemlerinin teknik özellikleri

| | | |
|---|---|---|
| Fiziksel Sunucu Özellikleri | İşlemci | 2 x Intel Xeon 5118 |
| | RAM | 64GB |
| | Chipset | Intel C621 Chipset |
| | Disk | 2 x 100GB RAID 1 (OS)<br>4 x 200GB RAID 5 (Data) |

## 3.3. VERİ TABANI ŞEMASININ BELİRLENMESİ

İlişkisel ve ilişkisel olmayan veri tabanlarının kıyaslanacağı bu çalışmada, Northwind veri tabanı kullanılmıştır. Northwind veri tabanı: Veri tabanı sistemlerini yeni öğrenenlerin, SQL sorgularında kendini geliştirmek isteyenlerin veya kritik sorgularını denemek isteyenlerin sıklıkla tercih ettiği Microsoft'un sağlamış olduğu ücretsiz, içerisi dolu bir veri tabanıdır [36]. Kıyaslama çalışmasını yapabilmek için biri MySQL diğeri ise MongoDB veri tabanı olmak

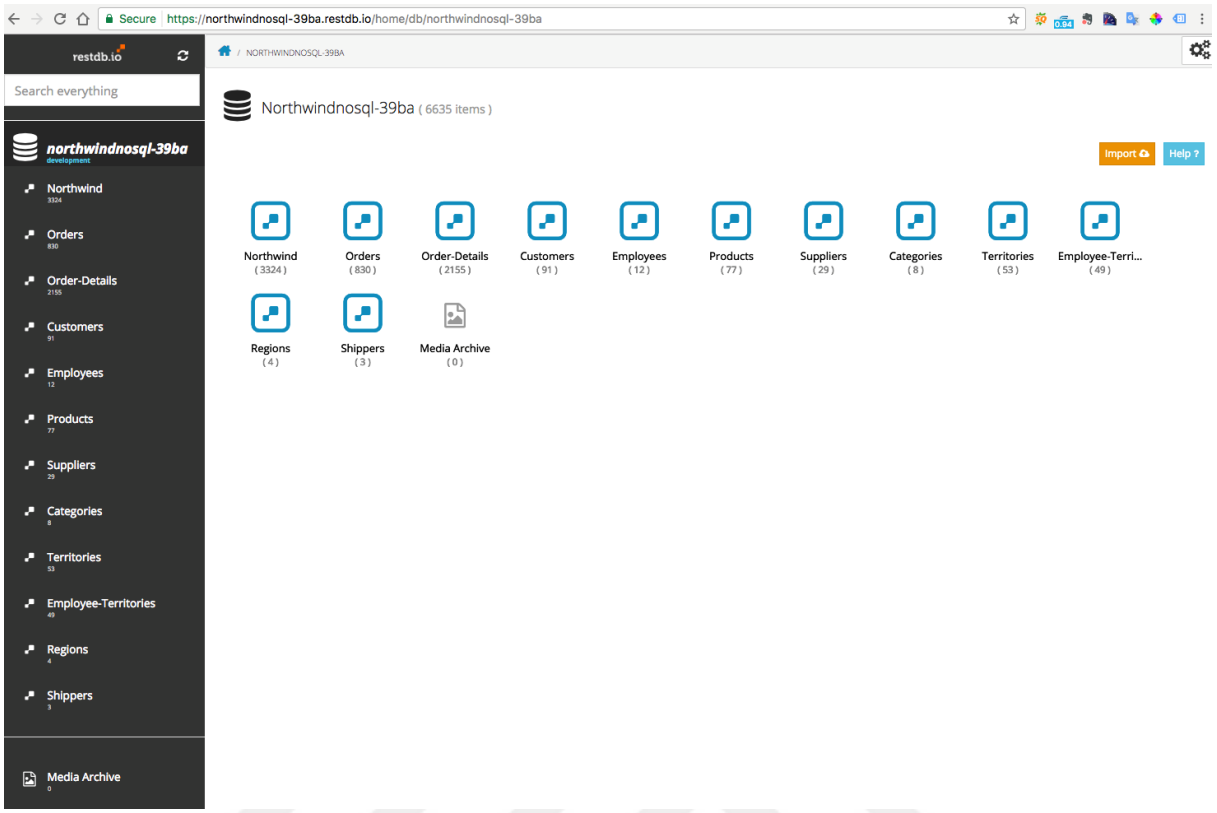
üzere iki adet aynı şemaya sahip veri tabanı tasarlanmıştır. İki veri tabanı arasında eşit şartları sağlamak adına ilişkisel veri tabanı tasarımında normalizasyon adımları uygulanmıştır. Tasarlanan ilişkisel veri tabanı şeması Şekil 3.1'de gösterilmiştir.




**Şekil 3.1:** Kullanılan ilişkisel veri tabanı şeması.

İlişkisel olmayan veri tabanlarında bir şema oluşturulmasına gerek duyulmamaktadır. İlişkisel olmayan veri tabalarında ilişkisel veri tabanı yönetim sistemlerden farklı olarak, koleksiyon yapıları çok daha esnektir. Koleksiyonlara yeni bir alan eklemesi veya mevcut bir alanın koleksiyondan çıkarılması sistemde bir sorun yaratmaz. Bu sistemlerde her doküman ayrı bir nesnedir ve kendine has bir yapısı olabilmektedir. Aynı zamanda bir tablo bir başka alt tablo da içerebilir. Bu alt tablolar ilişkisel veri tabanlarındaki tablolar arası ilişkiler ile benzer özellik taşımaktadır [34].

Yapılan çalışmayla, Northwind veri tabanı ilişkisel olmayan bir veri tabanı olan MongoDB sisteminde tekrardan tasarlanmıştır. Şekil 3.2'de aynı uygulama için hazırlanan ilişkisel olmayan veri tabanı şeması gösterilmektedir.




**Şekil 3.2:** İlişkisel olmayan veri tabanı şeması.

## 3.4. KULLANILACAK SORGULARIN BELİRLENMESİ

Bu çalışmada, farklı veri tabanı sistemlerinde karşılaşılabilecek veri tabanı sorgularını örneklendirmek adına, her iki veri tabanı için de aynı koleksiyonu üretecek dört farklı sorgu oluşturulmuştur. Farklı sorgu çeşitlerinin kullanılmasının nedeni; veri tabanlarında çalıştırılacak sorgu çeşitliğine, veri tabanlarının verdiği tepkiyi ölçümlemektir.

İlişkisel veri tabanı için; birinci sorgu olarak sadece “SELECT” cümlesi içeren basit bir sorgu oluşturulmuştur. İkinci sorguda tablo birleştirmelerini test etmek için birinci sorguya göre kısmen daha karışık “INNER JOIN” cümlesi içeren bir sorgu oluşturulmuştur. Üçüncü sorguda ise iç içe “SELECT” cümlesi içeren karmaşık bir sorgu oluşturulmuştur. Son olarak, dördüncü sorgu için hem “SELECT” hem “INNER JOIN” hem de iç içe “SELECT” cümlesi içeren daha karmaşık bir sorgu oluşturulmuştur.

İlişkisel olmayan veri tabanı sistemlerinde daha önce de belirtildiği gibi ilişkisel veri tabanı sistemlerinde olan “JOIN” türevi yapılara karşılık gelen yapılar bulunmamaktadır. Bu farklı

mimari göz önüne alındığında çalıştırılacak sorgular sadece bir koleksiyon üzerinde çalışmaktadır. Aşağıda MySQL veri tabanı için kullanılan SQL sorguları ile birlikte MongoDB veri tabanı sistemi için kullanılan sorgu karşılıkları şekiller ile gösterilmiştir.

```sql
SELECT * FROM Employees
WHERE FirstName = 'Omer'
```

**Şekil 3.3:** SELECT cümlesi içeren basit MYSQL sorgusu.

```
db.Employees.find({
  "FirstName":"Omer"
});
```

**Şekil 3.4:** SELECT cümlesi içeren basit MongoDB sorgusu.

```
SELECT o.* FROM Orders o
INNER JOIN Customers c on c.CustomerID = o.CustomerID
WHERE c.CompanyName = 'İstanbul Üniversitesi - Cerrahpaşa'
```

**Şekil 3.5:** INNER JOIN cümlesi içeren MYSQL sorgusu.

```
db.Orders.find({
  "Costomers.CompanyName":"İstanbul Üniv... - Cerrahpaşa"
});
```

**Şekil 3.6:** INNER JOIN cümlesi içeren MongoDB sorgusu.

```mysql
SELECT * FROM Orders
WHERE OrderID in (
  SELECT OrderID FROM Orders o
  WHERE o.ShipCountry = 'TR'
)
```

**Şekil 3.7:** Birden çok SELECT cümlesi içeren MYSQL sorgusu.

```mongodb
db.Orders.find({
  "OrderID":{
    "$in": ["Orders.ShipCountry = 'TR'"]
  }
});
```

**Şekil 3.8:** Birden çok SELECT cümlesi içeren MongoDB sorgusu.

```sql
SELECT o.* FROM Orders o
INNER JOIN OrderDetails od on od.OrderID = o.OrderID
WHERE od.OrderID in (
  SELECT o.OrderID FROM Orders o
  INNER JOIN Customers c on c.CustomerID = o.CustomerID
  WHERE c.CompanyName = 'İstanbul Üniversitesi - Cerrahpaşa'
)
```

**Şekil 3.9:** Birden çok SELECT ve INNER JOIN cümlesi içeren MYSQL sorgusu.

```
db.Orders.find({
  "$where": "this.OrderDetails.OrderID ==
              this.Orders.OrderID",
  "OrderID": {
    "$in": ["this.OrderID = Orders.OrderID"]
  },
  "Customer.CompanyName" = "İstanbul Üniv... - Cerrahpaşa"
});
```

**Şekil 3.10:** Birden çok SELECT ve INNER JOIN cümlesi içeren MongoDB sorgusu.

## 3.5. KARŞILAŞTIRMA METRİKLERİNİN BELİRLENMESİ

Veri tabanlarının karşılaştırılması işleminde, çalıştırılan sorguların zamanı hassas bir şekilde ölçülmüş ve kayıt altına alınmıştır. CPU üzerinde sorgunun çalışması süresi hesaplanarak performans sonuçları elde edilmiştir.

Yapılan çalışmada veri tabanı yönetim sistemlerinin farklı sorgu kalıplarına göre gösterdikleri performans incelenmiştir. Daha sonrasında ilişkisel ve ilişkisel olmayan veri tabanı sistemleri için ekleme ve silme performansları test edilmiştir. Yapılan test çalışmasında zaman ve kayıt sayısı parametreleri kullanılmıştır. İlişkisel ve ilişkisel olmayan veri tabanlarından elde edilen sonuçlar şekillerle gösterilmiştir. Ortaya çıkan değerler veri tabanı performansı olarak analiz edilmiştir.

Yapılan çalışmada daha önce belirtilen koşullar altında ilişkisel ve ilişkisel olmayan veri tabanlarının detaylı olarak karşılaştırılabilmesi için çeşitli durumlar oluşturulmuştur. Ölçüm için kullanılan kayıt sayıları 1000 ve 1000'in katları şeklinde test edilmiş, oluşan sonuçlar doğrusala yakın bir grafik oluşturduğu için, ölçüm 5000 kayıt ile sınırlandırılmıştır.

# 4. BULGULAR

## 4.1. SORGU PERFORMANS DEĞERLERİNİN BELİRLENMESİ

Sorgularının performans testleri bittikten sonra, kayıt ekleme ve silme performansları da ayrıca test edilmiştir. Ölçüm için kullanılan kayıt sayıları 5000 ve 5000'in katları şeklinde test edilmiş, oluşan sonuçlar doğrusala yakın bir grafik oluşturduğu için, ölçüm 25000 kayıt ile sınırlandırılmıştır.

Her bir testten önce fiziksel sunucu kapatılıp açılarak sorguların ön bellekten gelmediğinden emin olunmuştur. Her test sonunda sorguların çalışması sonucu alınan değerler hesaplanarak raporlanmıştır. Çalıştırılan bu sorgular MongoDB ve MySQL veri tabanı sistemleri için ayrı ayrı, eşit koşullar altında tekrarlanmıştır. Tablo 4.1'de ve Tablo 4.2'de sorgulara ait alınan sonuçlar belirtilmiştir.

**Tablo 4.1:** Sorgular çalıştırıldıktan sonra bulunan değerler.

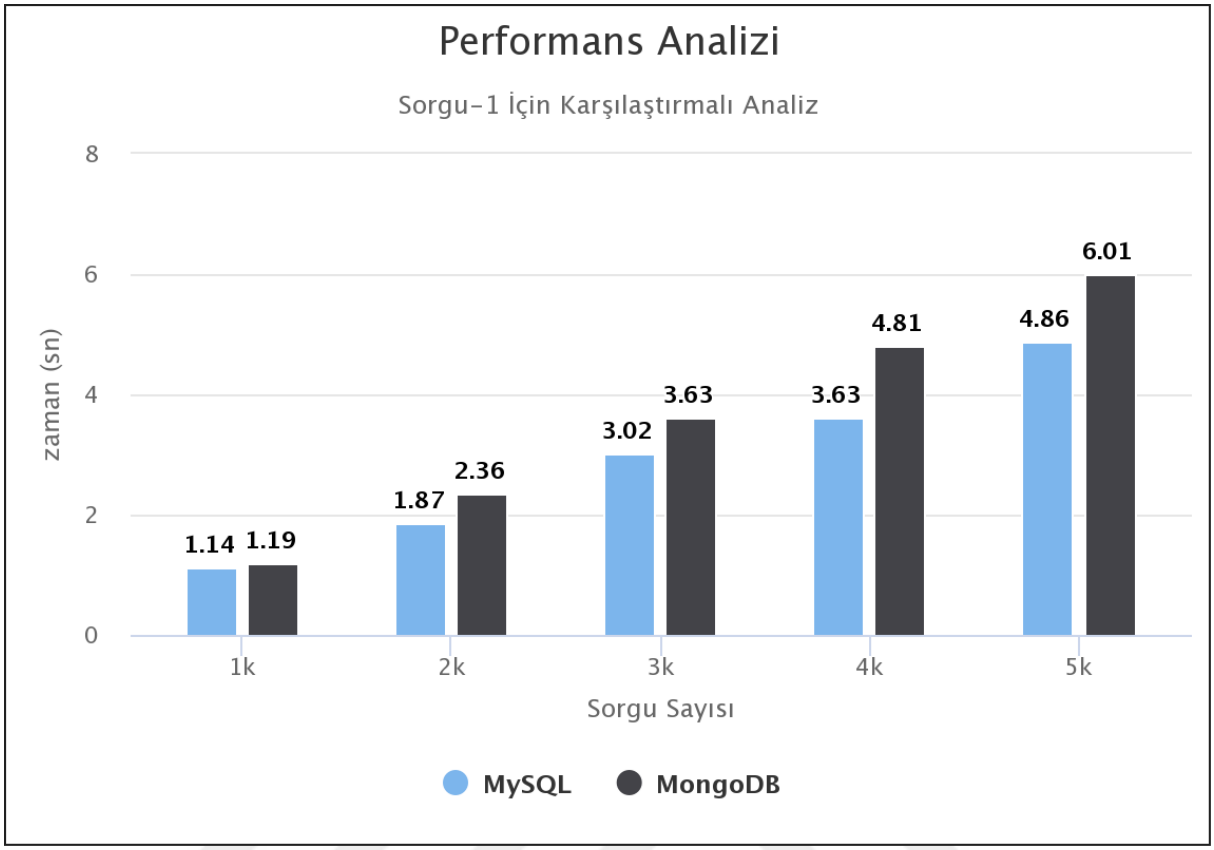
| | | Kayıt Sayıları | | | | |
|---|---|---|---|---|---|---|
| | | 1000 | 2000 | 3000 | 4000 | 5000 |
| MySQL | Sorgu 1 | 1.14 | 1.87 | 3.02 | 3.63 | 4.86 |
| MongoDB | Sorgu 1 | 1.19 | 2.36 | 3.63 | 4.81 | 6.01 |
| MySQL | Sorgu 2 | 3.10 | 6.09 | 9.14 | 12.40 | 15.32 |
| MongoDB | Sorgu 2 | 2.26 | 5.16 | 7.73 | 10.21 | 11.62 |
| MySQL | Sorgu 3 | 1.03 | 1.92 | 3.06 | 4.29 | 5.23 |
| MongoDB | Sorgu 3 | 1.86 | 3.75 | 5.96 | 8.04 | 10.07 |
| MySQL | Sorgu 4 | 18.63 | 61.32 | 121.86 | 172.29 | 223.36 |
| MongoDB | Sorgu 4 | 3.85 | 7.27 | 10.96 | 14.79 | 19.75 |

**Tablo 4.2:** Ekleme ve silme komutları çalıştırıldıktan sonra bulunan değerler.

| | | Kayıt Sayıları | | | | |
|---|---|---|---|---|---|---|
| | | 5000 | 10000 | 15000 | 20000 | 25000 |
| MySQL | Ekleme | 8.48 | 15.23 | 26.81 | 36.78 | 43.08 |
| MongoDB | Ekleme | 3.85 | 5.96 | 7.48 | 14.79 | 18.20 |
| MySQL | Silme | 9.24 | 18.08 | 24.43 | 38.59 | 44.43 |
| MongoDB | Silme | 11.73 | 22.45 | 37.10 | 44.26 | 56.34 |

## 4.2. PERFORMANS ANALİZİ

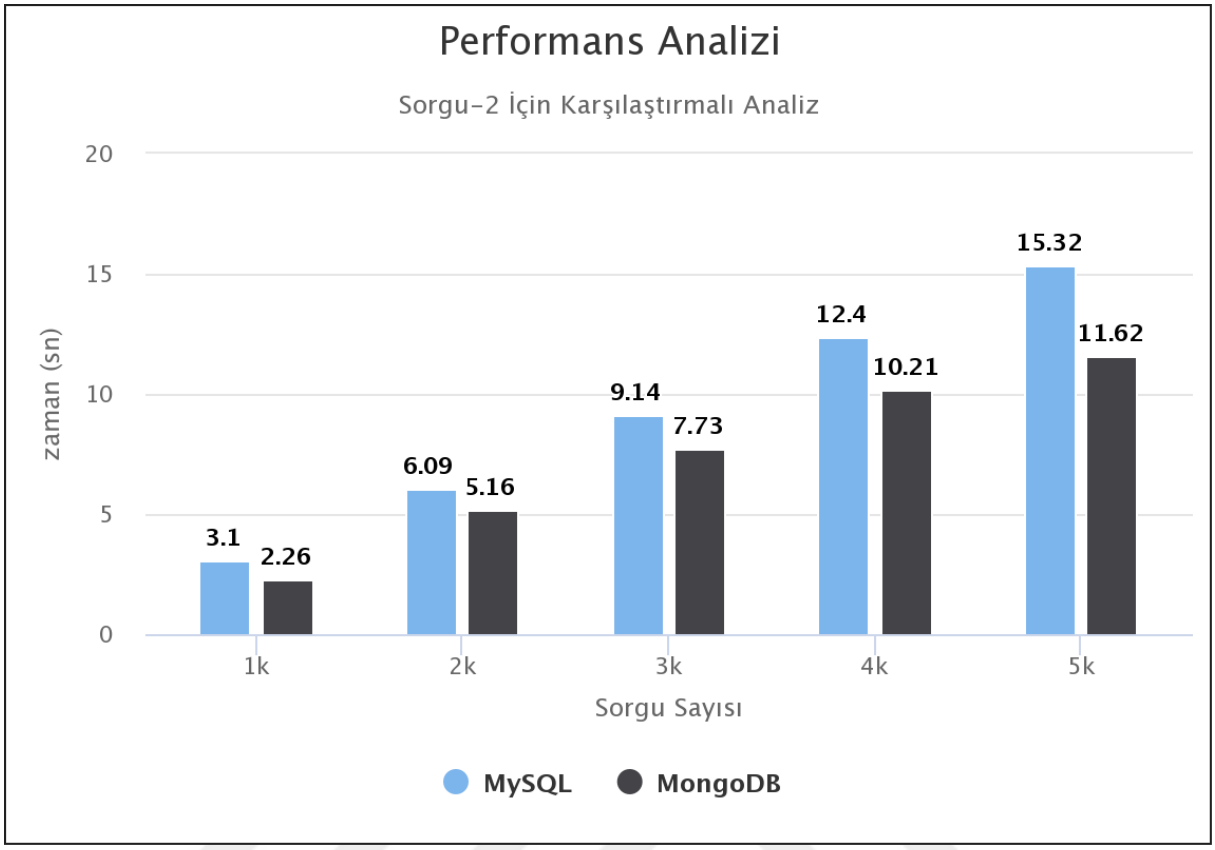
Yapılan çalışmada birinci sorgu olarak adlandırılan “SELECT” cümlesi içeren basit sorgu ile MySQL ve MongoDB veri tabanlarına karşılaştırma testi uygulanmıştır. Elde edilen sonuçlar Şekil 4.1’de gösterilmiştir.




**Şekil 4.1:** Sorgu-1 için karşılaştırılmalı performans analizi.

Şekil 4.1'de görüldüğü gibi, "SELECT" cümlesi içeren basit sorgu çalıştırıldığında MySQL veri tabanı sisteminin, MongoDB veri tabanı sisteminin üzerinde bir performans gösterdiği açıkça görülmektedir. Bu durum, ilişkisel olmayan veri tabanlarının hız avantajının tablo birleştirmeden yapılan sorgulamalarda geçerli olmadığını göstermektedir. Öyle ki, bu tip sorgularda ilişkisel veri tabanı sistemlerinin %15-20 oranında daha hızlı yanıt verdiği görülmektedir.

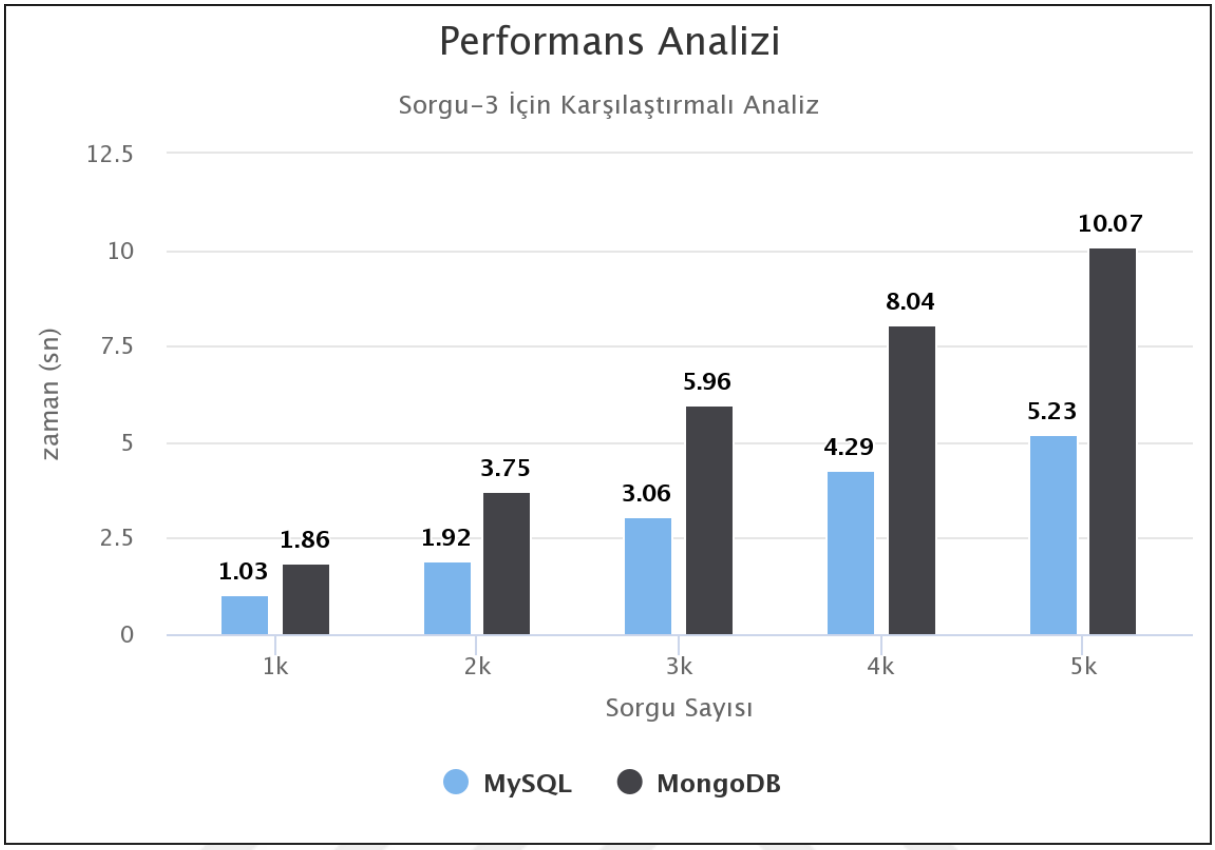
Yapılan çalışmada ikinci sorgu olarak adlandırılan "INNER JOIN" cümlesi içeren sorgu ile MySQL ve MongoDB veri tabanlarına karşılaştırma testi uygulanmıştır. Elde edilen sonuçlar Şekil 4.2'de gösterilmiştir.




**Şekil 4.2:** Sorgu-2 için karşılaştırılmalı performans analizi.

Şekil 4.2 'de görüldüğü gibi, "INNER JOIN" cümlesi içeren sorgu çalıştırıldığında MongoDB veri tabanı sisteminin, MySQL veri tabanı sisteminin üzerinde bir performans gösterdiği açıkça görülmektedir. Bu durum, ilişkisel olmayan veri tabanlarının ilişkisel olan veri tabanlarına göre tablo birleştirme gerektiren sorgularda %10-15 oranında daha performanslı çalıştığını göstermektedir.

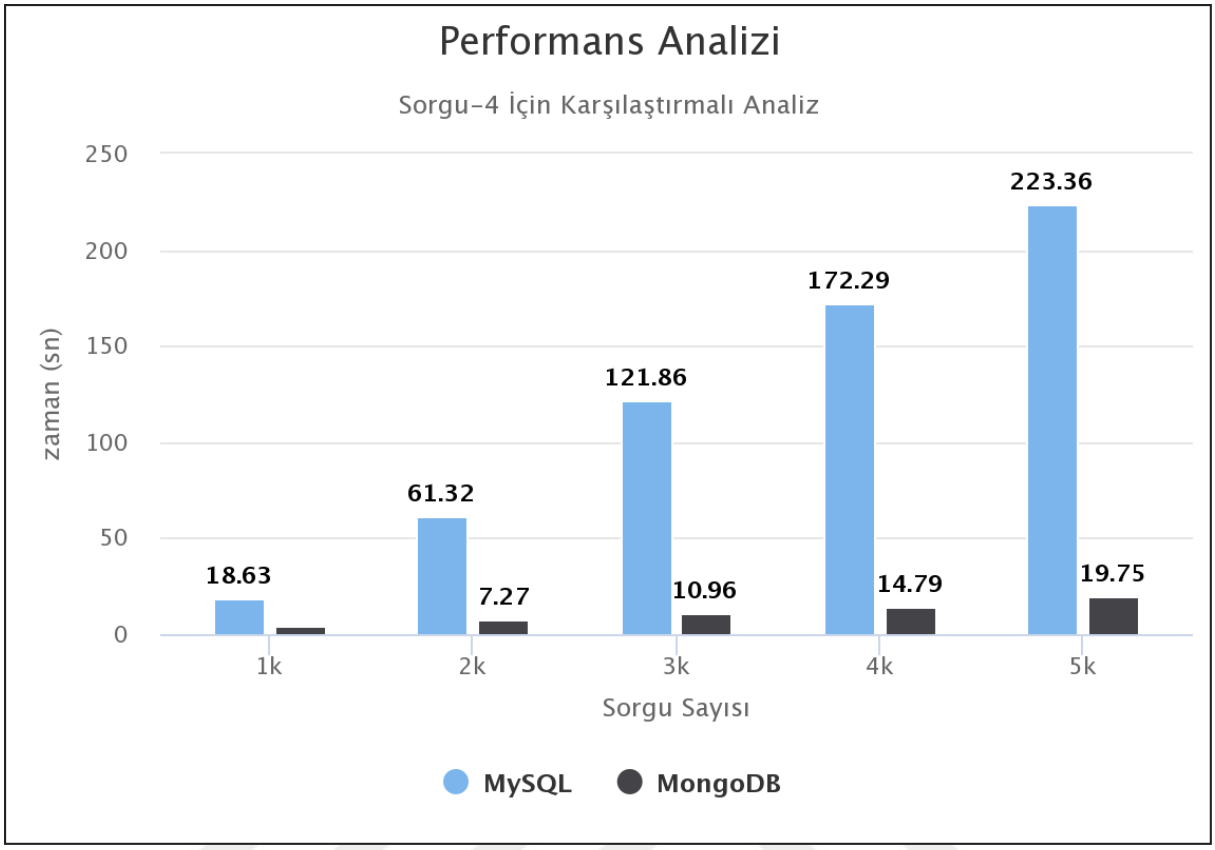
Yapılan çalışmada üçüncü sorgu olarak adlandırılan iç içe "SELECT" cümlesi içeren sorgu ile MySQL ve MongoDB veri tabanlarına karşılaştırma testi uygulanmıştır. Elde edilen sonuçlar Şekil 4.3'de gösterilmiştir.




**Şekil 4.3:** Sorgu-3 için karşılaştırmalı performans analizi.

Şekil 4.3'de görüldüğü gibi, iç içe "SELECT" cümlesi içeren kısmi karmaşık bir sorgu çalıştırıldığında MySQL veri tabanı sisteminin, MongoDB veri tabanı sisteminin üzerinde bir performans gösterdiği açıkça görülmektedir. Bu durum, ilişkisel olmayan veri tabanlarının hız avantajının tablo birleştirmeden yapılan sorgulamalarda geçerli olmadığını bir kez daha göstermektedir. Tablo birleştirme işlemine ihtiyaç duymadan yapılan kısmi karmaşık sorgularda ilişkisel veri tabanının ilişkisel olmayan veri tabanına göre %90-100 oranında daha performanslı çalıştığı görülmektedir.

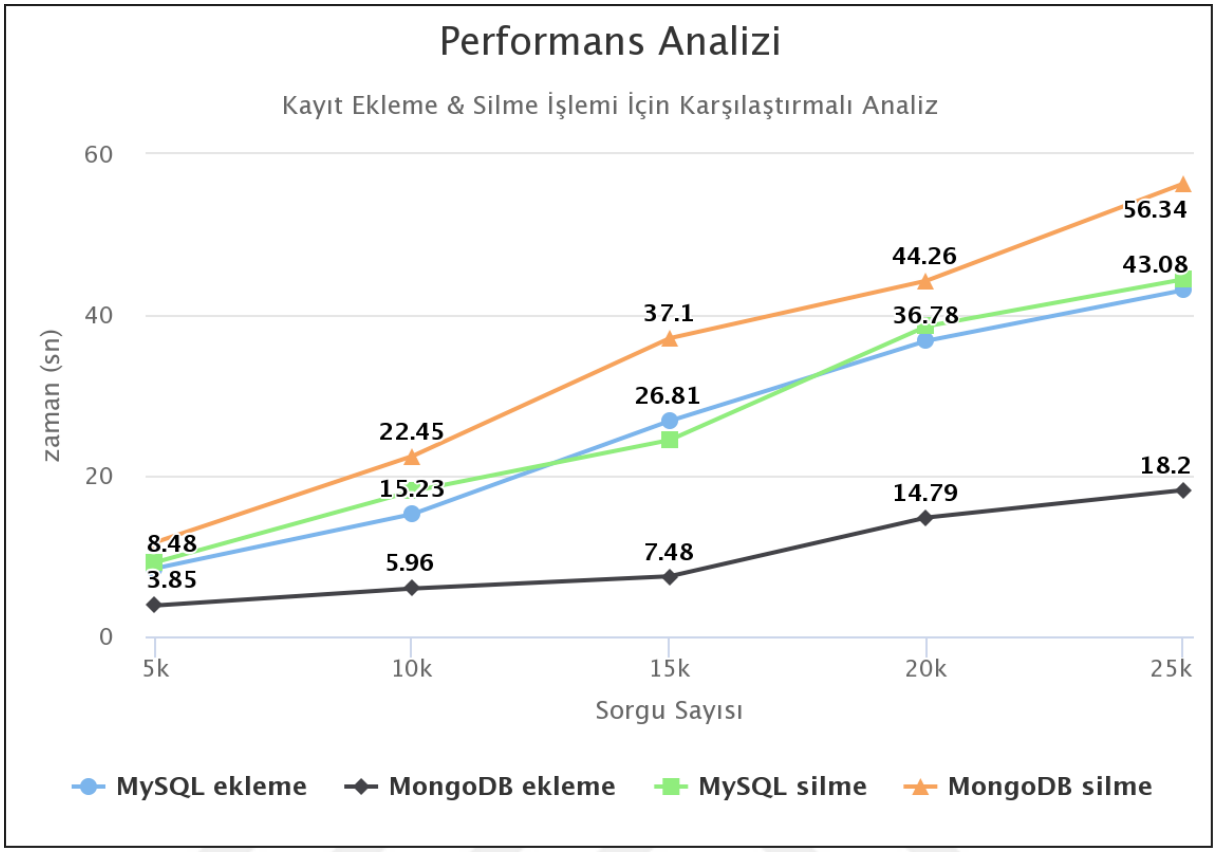
Yapılan çalışmada dördüncü sorgu olarak adlandırılan birden çok "SELECT" ve "INNER JOIN" içeren karmaşık sorgu ile MySQL ve MongoDB veri tabanlarına karşılaştırma testi uygulanmıştır. Elde edilen sonuçlar Şekil 4.4'de gösterilmiştir




**Şekil 4.4:** Sorgu-4 için karşılaştırmalı performans analizi.

Şekil 4.4'de görüldüğü gibi, birden çok SELECT ve INNER JOIN içeren karmaşık sorgu çalıştırıldığında MongoDB veri tabanı sisteminin, MySQL veri tabanı sisteminin oldukça üzerinde bir performans gösterdiği açıkça görülmektedir. MongoDB veri tabanı sisteminin sorguları işleme ve sonlandırma sürelerinin oldukça düşük ve istikrarlı olduğu da ayrıca görülmektedir.

Yapılan çalışmada son olarak ilişkisel veri tabanına ve ilişkisel olmayan ver tabanına ayrı ayrı kayıt ekleme ve silme testi uygulanmıştır. Elde edilen sonuçlar Şekil 4.5'de gösterilmiştir




Şekil 4.5: Kayıt ekleme ve silme işlemleri için karşılaştırmalı performans analizi.

Şekil 4.5'de görüldüğü gibi, MongoDB veri tabanı sisteminin kayıt ekleme işlemlerinde, MySQL veri tabanı sistemine göre çok daha hızlı yanıt verdiği anlaşılmaktadır. Bu fark 5000 sorgu sayısı ile az olsa da 25000 sorgu sayısına çıkıldığında daha beliğin olarak açığa çıkmaktadır. Kayıt silme işlemi, silinecek verinin bulunması ve daha sonra silinmesi adımları içerdiği için genel olarak kayıt ekleme işleminden fazla zaman alması beklenen bir değerdir. Kayıt silme işlemlerinde MySQL veri tabanı sisteminin MongoDB veri tabanı sistemine göre %10-15 daha iyi bir performans gösterdiği sonucu çıkartılabilir. Ekleme ve silme işlemlerinde bu farklılığın olmasının sebebi olarak iki farklı sistemin altyapısal ve operasyonel farklılıkları gösterilebilir.

# 5. TARTIŞMA VE SONUÇ

Her geçen gün daha da ilerleyen teknoloji etkisini veri tabanı yönetim sistemlerinde de ciddi şekilde hissettirmiştir. Sürekli büyüyen veriler hem ciddi performans hem de ciddi maliyet problemleri ortaya çıkarmıştır. Veri tabanı yönetimi noktasında yıllardır alternatifsiz olarak kullanılan ilişkisel veri tabanı yönetim sistemleri yetersiz kalmaya başlamıştır. Bu noktada çözüm olarak ilişkisel olmayan veri tabanı yönetim sistemleri (NOSQL) ortaya çıkmıştır. Yıllardır geliştiricilerin alışageldiği SQL dilini kullanmayan ve tutarlılık gibi bazı problemlerle ortaya çıkan bu teknoloji gün geçtikçe veri tabanı pazarındaki payını arttırmaktadır.

Günümüzde işletmelerin ve kurumsal firmaların çoğu ilişkisel veri tabanı yönetim sistemlerini kullanmaktadır. Bu ilişkisel veri tabanı yönetim sistemleri işletme ve kurumlara ciddi sunucu maliyetleri yüklemektedir. Yatay büyüme ile ölçeklenebilen, ilişkisel olmayan veri tabanları maliyetleri düşürmekle birlikte performans açısından da çok büyük avantajlar sağlayacaktır. Fakat bu noktada ilişkisel olmayan veri tabanı yönetim sistemlerinin dezavantajları da göz ardı edilmemelidir. ACID prensipleri henüz tam anlamıyla desteklemeyen ilişkisel olmayan veri tabanlarında veri tutarlılığı problemleri yaşanabilmektedir. Ayrıca ilişkisel veri tabanları, ilişkisel olmayan veri tabanlarında göre uzun yıllardır kullanıldığı için birçok raporlama, analiz ve performans araçlarına sahiptir.

Bu tez çalışması kapsamında yapılan testlerde, farklı sorgu tiplerinde seçilen her bir veri tabanının nasıl performans gösterdiği ortaya konmuştur. Ortaya çıkan performans değerleriyle veri tabanı yönetim sistemi seçimi noktasında destek olunması amaçlanmıştır. Veri ekleme performansı değerlendirildiğinde MongoDB, 3-4 kat daha iyi performans göstermiştir. Veri silme işleminde ise MYSQL her zaman üstün performans sergilemiştir. Diğer yandan seçilen dört farklı sorgu işleminin analizleri incelendiğinde, basit “select” komutu içeren sorgularda ve tablo birleştirme yapılmadan yani ilişkisel veri içermeyen sorgularda MYSQL daha iyi performans göstermiştir. MongoDB ise sorgu karmaşıklığı arttıkça ve birden fazla tablo ile birleştirme yapılarak veri getiren sorgularda çok daha üstün bir performans göstermiştir.

Yapılan karşılaştırma ve performans analizleri sonucunda, veri tabanı kullanacak bir organizasyon için kullanacağı uygulama kapsamında veri yapısının yapısal olup olmadığı, tahmini olarak ne kadar büyüklükte veri depolayacağı, veri artış hızı doğrultusunda yatay ölçeklendirmeye ihtiyaç duyup duymayacağı, veri tutarlığı konusundaki net duruşu ve birden fazla tablo ile sorgulama yapacak işlem sıklığı göz önünde bulundurularak veri tabanı yönetim sistemi seçimi konusunda karar alınabilir.

İlişkisel olmayan veri tabanlarının ne zaman kullanılacağı hakkında standart kurallar yoktur. Bu yüzden bu tez kapsamında yapılan karşılaştırma ve performans analizleri göz önünde bulundurularak geliştirilecek proje ihtiyaçlarına göre uygun veri tabanı yönetim sistemini seçmek mümkündür. Ayrıca geliştirilecek proje büyüklüğüne göre proje belirli parçalara bölünerek ilişkisel ve ilişkisel olmayan veri tabanlarının birlikte kullanılabileceği heterojen bir sistem de geliştirilebilir. Proje kapsamında yüksek performans gerektiren ve yapısal veriye ihtiyaç duyulmayan bölümlerde ilişkisel olmayan veri tabanları tercih edilebilir.

Gelecekte yapılacak çalışmalar için, bu tez kapsamı farklı veri tabanları da eklenerek performans analizleri genişletilebilir. Bu çalışma kapsamındaki performans analizleri önbellek göz ardı edilerek yapılmıştır, önbellek kullanımı ile birlikte test sonuçları tekrar değerlendirilebilir. Ayrıca çalışma bir sonraki aşamaya taşınıp ilişkisel bir veri tabanından, ilişkisel olmayan bir veri tabanına geçiş işlemleri yapılabilir.

# KAYNAKLAR


[1]. Diogo, M., Cabral, M., Bernardinho, J., 2019, Consistency Models of NoSQL Databases, future internet, 11(2), 43.

[2]. Khan, W., ahmed, E., Shahzad, W., 2017, Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases, International Journal of Advanced Computer Science and Applications, 8, 523-530.

[3]. Nguyen, N. H., 2016, An application-oriented comparison of two NoSQL database systems: MongoDB and VoltDB, Yüksek Lisans Tezi, Hamburg University, Faculty of Engineering and Computer Science.

[4]. Al-Saeedi, B., 2016, Factors influencing the database selection for B2C web applications, Yüksek Lisans Tezi, Technische Universitat München, Computational Science and Engineering.

[5]. Niyizamwiyitira, C., Lundberg, L., 2017, Performance Evaluation of SQL and NOSQL Database Management Systems in a Cluster, International Journal of Database Management Systems, 9(6), 1-24.

[6]. Veronika, A., Bernardinho, J., Furtado, P., 2014, Which NoSQL Database? A Performance Overview, Research Online Publishing, 1(2), 17-24.

[7]. Öztürk, S., Atmaca, H.E., 2017, İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi, Bilişim Teknolojileri Dergisi, 10(2), 199-209.

[8]. Gökşen, Y., Aşan, H., 2015, Veri Büyüklüklerinin Veritabanı Yönetim Sistemlerinde Meydana Getirdiği Değişim: NOSQL, Bilişim Teknolojileri Dergisi, 8(3), 125-131.

[9]. Kumar, L., Rajawat, S., Joshi, K., 2015, Comparative analysis of NoSQL (MongoDB) with MySQL Database, International Journal Of Modern Trends in Engineering and Research, 2(5), 120-127.

[10]. Dumanlı, B., 2018, En çok kullanılan ilişkisel ve NOSQL veritabanı yönetim sistemlerinin performans karşılaştırması, Yüksek Lisans Tezi, Trakya Univesitesi, Fen Bilimleri Enstitüsü.

[11]. Shwaysh, M., 2018, Security and Performance Comparison, Yüksek Lisans Tezi, Çankaya Univesitesi, Fen Bilimleri Enstitüsü.

[12]. Sethi, B., Mishra, S., Patnaik, P., 2014, A Study of NoSQL Database, International Journal of Engineering and Research & Technology, 3(4), 1131-1135.

[13]. Yılmaz, M., 2009, Enformasyon ve Bilgi Kavramları Bağlamında Enformasyon Yönetimi ve Bilgi Yönetimi, Ankara Üniversitesi Dil ve Tarih-Coğrafya Fakültesi Dergisi, 49(1), 97-101.

[14]. Jeffery, K., 2019, Data is the New Oil, https://indico.cern.ch/event/313634/attachments/600436/826361/Jeffery_Data_is_the_new_Oil.pdf, [Ziyaret Tarihi: 3 Ocak 2019].

[15]. Berg, K., Seymour, T., Goel, R., 2013, History Of Databases, International Journal of Management & Information Systems, 17(1), 29-36.

[16]. Mirzoev, T., Brockman, C., 2013, Sap Hana and Its Performance Benefits, i-manager's Journal on Information Technology, 2(1), 13-21.

[17]. Rai, P.K., 2015, Studies and Analysis of Popular Database Models, International Journal of Computer Science and Mobile Computing, 4(5), 834-838.

[18]. E. Önder, 2015, Yönetim Bilişim Sistemleri Kapsamında Web Tabanlı İlişkisel Veri tabanı Yönetim Sistemleri ve Bir Uygulama, Yüksek Lisans Tezi, İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü.

[19]. Vural, Y., Veritabanı Yönetim Sistemleri Güvenliği:Tehditler ve Korunma Yöntemleri, Politeknik Dergisi,13(2), 71-81.

[20]. Teorey, J., T., 1999, Database Modeling and Design, Morgan Kaufmann , Burlington, ISBN: 978-1558605008.

[21]. Alzahrani., H., 2016, Evolution of Object-Oriented Database Systems, Global Journals Inc., 16, 33-36.

[22]. Gönenç, A., 2004, Görsel veritabanı modeli 'İ.T.Ü. slayt arşivi veritabanı', Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[23]. Kalıpsız, O., 2001, Bilgisayar Veri Tabanı Sistemleri, Der Yayınları, İstanbul, ISBN: 9753532539.

[24]. Binani, S., Gutti, A., Upadhyay, S., 2016, SQL vs. NoSQL vs. NewSQL- A Comparative Study, Communications on Applied Electronics, 6(1), 43-46.

[25]. ScaleGrid, 2019 Database Trends, https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use, [Ziyaret Tarihi: 1 Mayıs 2019].

[26]. Alakoç Burma, Z., 2019, Veri Tabanı Yönetim Sistemleri, https://tr.scribd.com/document/81285535/veri-tabani-1-E-Kitap, [Ziyaret Tarihi: 3 Ocak 2019].

[27]. Hellerstein, M., Stonebaker, M., Hamilton, J., 2007, Architecture of a Database System, Foundations and Trends in Databases, 1(2), 141-259.

[28]. George, S., 2013, NOSQL - NotOnly SQL, International Journal of Enterprise Computing and Business Systems, 2(2).

[29]. Hossain, S.A., and Monuruzzaman, A.B.M., 2013, NoSQL Database: New Era of Databases for Big data Analytics-Classification, Characteristics and Comparison, International Journal of Database Theory and Application, 6, 3-4.

[30]. Cattell, R., 2009, Scalable SQL and NoSQL Data Stores, SIGMOD Record, 39(4), 13-14.

[31]. Pokorny., J., 2013, NoSQL databases: a stepto database scalabilityin web environment, International Journal of WebInformation Systems, 9, 69-82.

[32]. Microsoft, https://social.technet.microsoft.com/wiki/contents/articles/34195.nosql-dunyas-2-veri-tutarllk-modelleri-acid-vs-base-tr-tr.aspx, [Ziyaret Tarihi: 1 Nisan 2019].

[33]. Machado K., Kank, R., Sonawane, J., Maitra, S., 2017, A Comparative Study of ACID and BASE in Database Transaction Processing, International Journal of Scientific & Engineering Research, 8(5) 116-119.

[34]. Ghosh, K., Nath, A., 2016, NoSQL Database: An Advanced Way to Store, Analyze and Extract Results From Big Data, International Journal of Advance Research in Computer Science and Management Studies, 4, 205-207.

[35]. Jain, V., Upadhyay, A., 2017, MongoDB and NoSQL Databases, International Journal of Computer Applications, 167(10), 16-20.

[36]. Dyer, J.N., Rogers, C., 2015, Teaching Case Adapting the Access Northwind Database to Support a Database Course, Journal of Information Systems Education, 26(2), 85-98.

# ÖZGEÇMİŞ

| Kişisel Bilgiler | |
|---|---|
| Adı Soyadı | Ömer Coşkun |
| Doğum Yeri | Göynücek |
| Doğum Tarihi | 01.12.1989 |
| Uyruğu | ☑ T.C. ❒ Diğer: |
| Telefon | +90 544 372 91 98 |
| E-Posta Adresi | omer.coskun@istanbul.edu.trr |
| Web Adresi | |

| Eğitim Bilgileri | |
|---|---|
| **Lisans** | |
| Üniversite | Sakarya Üniversitesi |
| Fakülte | Mühendislik Fakültesi |
| Bölümü | Bilgisayar Mühendisliği |
| Mezuniyet Yılı | 2011 |

| Yüksek Lisans | |
|---|---|
| Üniversite | İstanbul Üniversitesi-Cerrahpaşa |
| Enstitü Adı | Lisansüstü Eğitim Enstitüsü |
| Anabilim Dalı | Bilgisayar Mühendisliği Anabilim Dalı |
| Programı | Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı |

| Makale ve Bildiriler |
|---|
| |