
GAZİ ÜNİVERSİTESİ
Gazi
1926

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

Serdar ÖZTÜRK

YÜKSEK LİSANS TEZİ

YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

29 Kasım 2017

Serdar ÖZTÜRK tarafından hazırlanan "İLİŞKİSEL VE İLİŞKİSEL OLMAYAN (NOSQL) VERİ TABANI YÖNETİM SİSTEMLERİ MİMARİ PERFORMANSININ YÖNETİM BİLİŞİM SİSTEMLERİ KAPSAMINDA İNCELENMESİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Yönetim Bilişim Sistemleri Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Hatice Ediz ATMACA

Endüstri Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Başkan: Prof. Dr. O. Ayhan ERDEM

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Üye: Yrd. Doç. Dr. Onur İŞBULAN

Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Sakarya Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Tez Savunma Tarihi: 29/11/2017

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

Doç. Dr. Bünyamin CİYLAN

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Serdar ÖZTÜRK

29/11/2017

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

(Yüksek Lisans Tezi)

Serdar ÖZTÜRK

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Kasım 2017

ÖZET

Günümüzdeki bilgi ve iletişim teknolojilerinde yaşanan değişim ve gelişime bağlı olarak, verilerin modellenmesi ve belli bir modelle ifade edilen verilerin saklanması veri tabanı kullanımını zorunlu kılmaktadır. Temel bir kurum rehberinden, orta ve büyük ölçekli işletmelerin kurumsal ve ticari bilgilerinin organize edilerek saklanmasına kadar farklı alanlarda veri modelleme ve depolanması bir zorunluluk haline gelmiştir. Verinin büyüklüğü, miktarı ve karmaşıklığı gibi etkenlere bağlı olarak farklı veri modelleme, veri depolama ve sorgulama yöntemleri geliştirilmiştir. Veri miktarındaki ciddi artış, uzun süredir ihtiyaçlara cevap veren ilişkisel veri tabanı sistemleri ile birlikte ilişkisel olmayan (NoSQL) sistemlerin de gelişmesine neden olmuştur. Geliştirilen birçok ilişkisel veri tabanı yanı sıra yakın geçmiş itibarıyla ortaya çıkan çok sayıdaki NoSQL veri tabanları, farklı özellikleri nedeni ile, organizasyonların yönetim bilgi sistemlerinde çalışan veri tabanı yöneticilerinin seçim yapmasını zorlaştırmaktadır. İlişkisel ve NoSQL veri tabanı özelliklerini iyi bilerek, hazırlanan uygulamaların niteliklerine en uygun veri tabanını seçmek gerek maliyet gerekse hız açısından oldukça önemlidir. Yapılan çalışmada işletmeler tarafından kullanılan veri tabanı yönetim sistemlerinin seçimi problemi üzerinde durulmuştur. Firmaların yönetim bilgi sistemlerinde çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanlarının farklı veri tabanı sorgu türleri karşısında gösterdikleri performans karşılaştırılmıştır. İşletmelere hangi durumda nasıl bir veri tabanı yönetim sistemi kullanmaları konusunda önerilerde bulunulmuştur.

Bilim Kodu	:	114611
Anahtar Kelimeler	:	İlişkisel veri tabanı, ilişkisel olamayan (NoSQL) veri tabanı, MongoDB
Sayfa Adedi	:	79
Danışman	:	Doç. Dr. Hatice Ediz ATMACA

THE EXAMINATION OF RELATIONAL AND NOSQL DATABASE MANEGEMENT SYSTEM'S ARCHITECTURAL PERFORMANCES IN TERMS OF MANAGEMENT INFORMATION SYSTEMS

(M. Sc. Thesis)

Serdar ÖZTÜRK

GAZİ UNIVERSITY

INSTITUTE OF INFORMATICS

November 2017

ABSTRACT

Changes and developments in information and communication technologies in modern-day, modeling of data and storage of data expressed in a certain model make it essential to use a database. The need for data modeling and storage in different areas arises, from a basic corporate directory to organizing and storing corporate and commercial information for medium and large-scale enterprises. Different data modeling, data storage and querying methods have been developed, depending on factors such as size, amount and complexity of the data. A significant increase in the amount of data has led to the development of nonrelational database systems (NoSQL) as well as relational database systems that respond to long-responding needs. Numerous NoSQL databases emerging in recent years as well as many developed relational databases make it difficult for database administrators, who work in the management information systems of organizations, to select with distinctive characteristics. In the work being done, it is emphasized on the problem of selection of database management systems that is used by the enterprises. Performance of relational and NoSQL databases against diverse types of database query types was compared with an application selected based on applications that are frequently developed in the management information systems of companies. The results of the tests were analyzed and the effects on the operational efficiency of the NoSQL database management systems were investigated by making radical changes in the applications developed by the management information systems of the firms or institutions.

Science Code : 114611
Key Words : Relational database, NoSQL database, MongoDB
Page Number : 79
Supervisor : Assoc. Prof. Dr. Hatice Ediz ATMACA

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, iki yıl boyunca değerli bilgilerini benimle paylaşan, güler yüzlü saygıdeğer danışman hocam; Doç. Dr. Hatice Ediz ATMACA'ya, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen arkadaşım Mustafa KARAKOÇ'a ve çalışma süresince tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek olan değerli eşim Zahide Selen ÖZTÜRK'e sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ ix

ŞEKİLLERİN LİSTESİ x

SİMGELER VE KISALTMALAR xii

1. GİRİŞ 1

2. LİTERATÜR ARAŞTIRMASI 5

3. BİLİŞİM SİSTEMLERİ VE VERİ TABANI TEKNOLOJİLERİ 9

3.1. Bilişim Sistemleri, Faaliyetleri ve Yaklaşımları 9

3.2. Veri Tabanı Tanımı, Tasarımı ve Optimal Performansı 17

3.3. İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Yönetim Sistemleri 34

3.3.1. İlişkisel veri tabanı yönetim sistemleri 35

3.3.2. İlişkisel olmayan (NoSQL) veri tabanı yönetim sistemleri 37

3.3.3. İlişkisel ve NoSQL veri tabanı yönetim sistemlerinin karşılaştırılması 45

4. UYGULAMA: İLİŞKİSEL VE İLİŞKİSEL OLMAYAN (NoSQL) VERİ TABANI YÖNETİM SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI 51

4.1. Seçilen Veri Tabanı Yönetim Sistemleri ve Seçim Nedenleri 52

4.2. Veri Tabanı Sunucu Sistemleri Özellikleri 53

4.3. Veri Tabanı Şeması 53

4.4. Veri Tabanı Sorguları 55

Sayfa

4.5. Ölçümler ve Ölçüm Metrikleri 57

4.6. Sonuçlar ve Analiz 59

5. SONUÇ VE ÖNERİLER 65

KAYNAKLAR 68

EKLER 75

EK-1. Test sonuçları 76

EK-1. (devam) Test sonuçları 77

EK-2. Konfigürasyon 78

ÖZGEÇMİŞ 79

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 3.1. Düz veri modeli örneği..... 19

Çizelge 3.2. Ana tablo (normalizasyon örneği) 28

Çizelge 3.3. Birinci normal form tablo örneği 28

Çizelge 3.4. 2NF ana tablo..... 29

Çizelge 3.5. 2NF servis tablosu..... 29

Çizelge 3.6. 3NF ana tablo..... 30

Çizelge 3.7. 3NF servis tablosu..... 30

Çizelge 3.8. 3NF şoför tablosu..... 30

Çizelge 3.9. BCNF ana tablo 31

Çizelge 3.10. BCNF servis tablosu 31

Çizelge 3.11. BCNF semt tablosu..... 31

Çizelge 3.12. BCNF şoför tablosu 31

Çizelge 4.1. Fiziksel ve sanal sunucu sistemleri teknik özellikleri 53

Çizelge 1.1. Sorgu 1 (sadece SELECT içeren basit sorgu) ortalama zaman çizelgesi..... 76

Çizelge 1.2. Sorgu 2 (JOIN içeren karmaşık sorgu) ortalama zaman çizelgesi 76

Çizelge 1.3. Sorgu 3 (JOIN içermeyen karmaşık sorgu) ortalama zaman çizelgesi 76

Çizelge 1.4. Sorgu 4 (JOIN içeren detaylı karmaşık sorgu) ortalama zaman çizelgesi 76

Çizelge 1.5. Veri ekleme ve silme işlemleri ortalama zaman çizelgesi 77

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 3.1. Bilişim sistemleri bileşenleri 10

Şekil 3.2. Temel bilişim sistemleri 12

Şekil 3.3. VT, VTYS ve VTS arasındaki ilişki ve işlevler 18

Şekil 3.4. Hiyerarşik veri tabanı model örneği 19

Şekil 3.5. Ağ veri model örneği 20

Şekil 3.6. İlişkisel veri model örneği 21

Şekil 3.7. Nesne yönelimli veri model örneği 22

Şekil 3.8. Nesne ilişkisel ileri model örneği 23

Şekil 3.9. Veri tabanı tasarımı aşamaları 26

Şekil 3.10. Normalizasyon algoritması 32

Şekil 3.11. Veri tabanları kullanım grafikleri 38

Şekil 3.12. Bazı NoSQL veri tabanlarının sınıflandırılması 39

Şekil 3.13. Zaman serisi uygulamalarında veri tabanı teknolojileri kullanım oranları 43

Şekil 3.14. Yatay ve dikey ölçekleme (scale-up ve scale-out) kavramları 45

Şekil 4.1. İlişkisel (MySQL) veri tabanı şeması 54

Şekil 4.2. NoSQL (MongoDB) veri tabanı şeması 55

Şekil 4.3. Sadece SELECT değimi içeren basit sorgu için SQL sorgu dili 56

Şekil 4.4. Sadece SELECT değimi içeren basit sorgu için MongoDB söz dizimi 56

Şekil 4.5. Sadece bir INNER JOIN içeren karmaşık sorgu için SQL sorgu dili 56

Şekil 4.6. Sadece bir INNER JOIN içeren karmaşık sorgu için MongoDB söz dizimi 56

Şekil 4.7. İç içe SELECT içeren karmaşık sorgu için SQL sorgu dili 57

Şekil 4.8. İç içe SELECT içeren karmaşık sorgu için MongoDB söz dizimi 57

Şekil **Sayfa**

Şekil 4.9. İç içe SELECT ve INNER JOIN içeren karmaşık sorgu için SQL sorgu dili 57

Şekil 4.10. İç içe SELECT ve INNER JOIN içeren karmaşık sorgu için MongoDB söz dizimi 57

Şekil 4.11. Sorgu 1 kullanılarak veri tabanı sistemlerinin performans karşılaştırması 59

Şekil 4.12. Sorgu 2 kullanılarak veri tabanı sistemlerinin performans karşılaştırması 60

Şekil 4.13. Sorgu 3 kullanılarak veri tabanı sistemlerinin performans karşılaştırması 61

Şekil 4.14. Sorgu 4 kullanılarak veri tabanı sistemlerinin performans karşılaştırması 62

Şekil 4.15. Veri tabanı sistemlerinin veri ekleme ve silme performansı kıyaslaması 63

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
ACID	Bölünmezlik, tutarlılık, izolasyon, dayanıklılık (Atomicy, concistancy, isolation, durability)
BCNF	Boyde-Codd normal form
DB	Veri tabanı **(**Database)
DBMS	Veri tabanı yönetim sistemi (Database management system)
İVTYS	İlişkisel veri tabanı yönetim sistemi
JSON	JavaScript nesne gösterimi (Javascript object notation)
LAN	Yerel alan ağı (Local area network)
M2M	Makineler arası iletişim (Machine to machine)
NF	Normal form
RDBMS	İlişkisel veri tabanı yönetim sistemi (Relation database management system)
SQL	Yapısal sorgulama dili (Structured query language)
VT	Veri tabanı
VTS	Veri tabanı sistemi
VTYS	Veri tabanı yönetim sistemi
YBS	Yönetim bilişim sistemleri
XML	Genişletilebilir işaretleme dili (Extensible markup language)

1. GİRİŞ

Bilgisayar ve iletişim teknolojilerinde yaşanan hızlı değişim ve gelişim her geçen gün daha fazla organizasyonu etkilemektedir. Belli başlı bir amaca ulaşmak için veri veya ham bilginin işlenerek ilgililere yarar sağlayacak biçime dönüştürülmüş hali olan bilgi, organizasyonlar tarafından sürekli daha kısa sürede erişilmek istenen etkili bir faktör haline gelmiş ve önemli bir rekabet unsuru oluşturmuştur. Bilgisayarların karar alma süreçlerinde etkin olarak kullanılması ile, "bilgi sistemleri" günümüzün revaçta olan konuları arasındaki yerini almıştır [1].

Yaşanan bu değişim ve gelişim, verilerin modellenmesini ve belli bir modelle ifade edilen verilerin saklanmasını dolayısıyla veri tabanı kullanımını zorunlu kılmaktadır. Temel bir kurum rehberinden, orta ve büyük ölçekli işletmelerin kurumsal ve ticari bilgilerinin organize edilerek saklanmasına kadar farklı alanlarda veri modelleme ve depolama gerekliliği ortaya çıkmaktadır. Verinin büyüklüğü, miktarı, karmaşıklığı ve tutarlılığı gibi etkenlere bağlı olarak farklı veri modelleme, veri depolama ve sorgulama yöntemleri gelişimi her geçen gün sürmektedir. Bu gelişim, yakın geçmişte ilişkisel olmayan (NoSQL) veri tabanı yönetim sistemlerini gün yüzüne çıkartmıştır.

Bu kapsamda, okuma ve yazma gibi işlemlerin yoğun olarak kullanıldığı veri tabanlarında ilişkisel veri tabanlarının yanı sıra ilişkisel olmayan (NoSQL) veri tabanı yönetim sistemleri de yoğun olarak kullanılmaya başlanmıştır [2]. Performans ve esneklik gibi öne çıkan özellikleri ile NoSQL veri tabanı yönetim sistemleri; büyük veri ile başa çıkmak zorunda olan eBay ve Amazon gibi dünyaca ünlü şirketler tarafından tercih edilebilir hale gelmiştir [3].

Geliştirilen birçok ilişkisel veri tabanı yanı sıra yakın geçmiş itibarıyla ortaya çıkan çok sayıdaki NoSQL veri tabanı, farklı özelliklerine bağlı avantaj ve dezavantajları nedeni ile, organizasyonların yönetim bilgi sistemlerinde çalışan veri tabanı yöneticilerinin hazırlanan/hazırlanacak uygulamalarda uygun bir veri tabanı yönetim sistemini seçebilmesi problemini doğurmuştur.

Bu çalışmada işletmeler tarafından kullanılan veri tabanı yönetim sistemlerinin seçimi problemi üzerinde durulmuştur. Öncelikle ilişkisel ve NoSQL veri tabanı yönetim sistemleri açıklanmıştır. NoSQL veri tabanlarının dört veri modeli tek tek ele alınarak işletmeler açısından güçlü ve zayıf tarafları ortaya konmuştur. Mevcut durumda firmaların kullandığı veri tabanı sistemlerinin belirtildiği bu çalışmada, ilişkisel ve NoSQL veri tabanlarının avantaj ve dezavantajları; yatay ve dikey ölçeklenebilirlik, ACID ve BASE kuralları, yapısal ve yapısal olmayan veri yapıları ile yazılan ve okunan şema parametrelerine göre ele alınmıştır. Son olarak firmaların yönetim bilgi sistemlerince çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanları performansı karşılaştırılmış, yeni geliştirilecek uygulamalarda kullanıcılara secim konusunda fikir verilerek NoSQL veri tabanı yönetim sistemlerinin sağladığı bilinen hız avantajının, firmaların mevcut uygulamalarda değişime gitmesini gerektirip gerektirmediği sorgulanmıştır.

1’inci bölüm giriş bölümüdür. Bu bölümde problem durumu tam olarak ortaya konmuş ve bu kapsamda yapılan çalışma hakkında bilgi verilmiştir.

2’nci bölümde anahtar kelimeler kullanılarak yapılan sorgular kapsamında konu ile alakalı olarak bugüne kadar hazırlanan makale ve tezlerin yer aldığı literatür araştırmasına yer verilmiş ve çalışmanın literatüre katkısı vurgulanmıştır.

3’üncü bölüm üç alt başlıktan oluşmaktadır. Öncelikle, bilişim sistemleri tanımı, temel bileşenleri, sınıflandırması ile bilişim sistemleri yaklaşımlarına yer verilmiştir. Sonrasında; veri tabanı, veri tabanı yönetim sistemi ve veri tabanı sistemi kavramları açıklanmış, veri tabanı modelleri grafik ve şekiller ile anlatılarak nihayet örnek bir veri tabanının nasıl tasarlanacağı detaylı olarak açıklanmıştır. Son alt başlıkta ise; öncelikle ilişkisel ve NoSQL veri tabanları özellikleri verilerek sınıflandırılması yapılmıştır. Sonrasında, NoSQL veri tabanlarının dört veri modeli tek tek ele alınarak işletmeler açısından güçlü ve zayıf tarafları ortaya konmuştur. Mevcut durumda firmaların kullandığı veri tabanı sistemlerinin belirtildiği bu bölümde, ilişkisel ve NoSQL veri tabanlarının avantaj ve dezavantajları; yatay ve dikey ölçeklenebilirlik, ACID ve BASE kuralları, yapısal ve yapısal olmayan veri yapıları ile yazılan ve okunan şema parametrelerine göre ele alınmıştır.

4’üncü bölümde; firmaların yönetim bilgi sistemlerince çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanları performansı karşılaştırılmıştır. Çalışmanın yapıldığı makina özellikleri, ölçüm metrikleri ve sorgulara yer verilen bu bölümde, elde edilen sonuçlar şekiller ile gösterilmiştir.

5’inci ve son bölüm sonuç ve öneriler bölümüdür. Bu bölümde yapılan testlerden elde edilen sonuçlar analiz edilmiş, firma veya kurumların yönetim bilgi sistemlerince geliştirilmiş olan uygulamalarda köklü değişiklikler yapılarak, NoSQL veri tabanı yönetim sistemlerinin sağladığı bilinen hız avantajının elde edilmesinin, işletme verimliliği üzerindeki etkileri sorgulanmıştır. Son olarak yapılan çalışma ile elde edilen sonuçlar kapsamında ileride yapılabilecek çalışmalar hakkında öneriler sunulmuştur.

2. LİTERATÜR ARAŞTIRMASI

Günümüze kadar değişen ihtiyaçlara cevap verebilmek için veri tabanı teknolojileri de gelişim göstermiştir. Veri miktarındaki ciddi artış, uzun süredir ihtiyaçlara cevap veren ilişkisel veri tabanı sistemleri ile birlikte NoSQL sistemlerin de gelişmesine neden olmuştur. Geliştirilen çok sayıdaki veri tabanı yapılarının farklı özellikleri, kullanıcıların seçim yapmasını zorlaştırmaktadır.

Bu bölümde, yapılan çalışmanın konusunu oluşturan ilişkisel veri tabanları ile NoSQL veri tabanlarını ihtiva eden çalışmaların literatürdeki gelişimleri incelenmiştir. Literatürde, ilişkisel ve NoSQL veri tabanlarının avantaj ve dezavantajlarına yer veren ve NoSQL veri tabanlarının en sık kullanılanlarını kendi arasında çeşitli parametrelere göre karşılaştıran tekniklere yer verildiği gözlemlenmiştir. Özellikle son yıllarda NoSQL veri tabanlarının yaygın olarak kullanılmaya başlanması ile çeşitli karşılaştırmalar içeren az sayıdaki çalışmalar dışında, kullanıcıların geliştirdikleri uygulamalarda kullanmak zorunda oldukları farklı özelliklerdeki veri tabanlarının seçimi problemi üzerinde yeterli çalışmaların yapılmadığı görülmüştür. Özellikle performansla ilgili yapılan çalışmalarda farklı sorgu türlerinin hıza olan etkisinin ele alındığı bir çalışmaya rastlanmamıştır. Bu çalışmanın literatüre katkısı açısından önceki çalışmalar ayrı ayrı incelenmiştir.

Literatürde yapılan bazı çalışmalar şöyledir

Orend [4], çalışmasında nesne ilişkisel veri tabanları yerine konulabilmesi için NoSQL veri tabanlarının analizi, sınıflandırılması ve yeteneklerine göre değerlendirilmesi üzerinde durmuştur. Hazırlamış olduğu çalışmada, NoSQL veri tabanlarını kendi içerisinde karşılaştırılmıştır.

Muhammad [5], çalışmasında NoSQL veri tabanları üzerinde bir değerlendirme yapmıştır. Çalışmasında ilişkisel ve NoSQL veri tabanları arasında güvenilirlik mukayesesi yapmıştır. Klasik ilişkisel veri tabanlarının büyük ve tüm verileri ölçeklemekte çok zorlandığını ve bu işlemlerin üstesinden gelmenin bir yolunun NoSQL veri tabanı yönetim sistemleri olduğunu belirtmiştir.

Hecht ve Jablonski [6], yayınlarında NoSQL veri tabanını seçmenin zorluğuna vurgu yapmış ve bu veri tabanlarının uygulanabilirlik, veri modelleri, sorgulama olasılıkları, eşzamanlılık gibi özelliklerini karşılaştırmıştır. Web üzerindeki verinin her geçen gün artmasını başarılı web siteleri (Facebook, Amazon, Google) için bir problem olarak değerlendirildiğini belirtmiştir. Petabyte'larca veri ile başa çıkmak, bu verileri okumak ve isteklere geciktirmeden cevap vermek gerektiğinde normalize olmuş veri modelleri ve ACID kuralları ile bu ihtiyaca cevap verebilmenin mümkün olmadığını, tablo birleştirmenin performansa negatif etki ettiğini vurgulamıştır.

Brad [7], yayımlanmış olan tezinde NoSQL veri tabanlarının veri tekilleştirmesi performansını araştırmıştır. Tekrarlayan verileri mümkün olduğunca tekilleştiren çalışmasında performans arttırmayı amaçlamıştır.

Roijackers [8], yüksek lisans tezinde SQL ve NoSQL veri tabanları arasında bölüştürülen verilerin getirmiş olduğu zorlukları aşmak amaçlı bir soyut veri tabanı geliştirmiştir.

Näsholm [9], çalışmasında NoSQL veri tabanlarının bağımsız şema ve yatay ölçekleme özelliklerinin bazı uygulamalarda veri yazma ve okumada problemlere yol açtığı üzerinde durmuştur. Günümüz modern uygulamaların gereksinimlerinin değiştiğini belirterek çok büyük web sitelerinin milyarlarca kez sayfaları hizmet olarak sunmak zorunda olduğunu ve kullanıcıların bu sayfalara hızlı biçimde erişmek istemesinin ilişkisel veri tabanları yönetim sistemleri ile tatmin edici bir şekilde karşılanamadığını vurgulamıştır.

Shermin [10], yayınlanmış olan çalışmasında NoSQL veri tabanlarındaki klasik rol tabanlı erişim kontrolünü geliştirerek bağlam bazlı soyut bir model sunmuştur. Sunduğu bu modelin güvenlik politikalarını değerlendirmiş ve çalıştığını belirtmiştir.

Benzer [11], tezinde Ontologlar tarafından kullanım kolaylığı sağlamasını amaçlayan bir web uygulaması geliştirilmesine yönelik çalışmalara yer vermiştir. Bu web uygulamasının geliştirilmesinde Scala, Play Framework 2, MongoDB ve Angular.js teknolojileri kullanılmıştır. Çok sayıda kullanıcının aynı anda kullanabilmesini sağlamak amacıyla ölçeklenebilir bir mimari kullanılmıştır.

Yar ve Tun [12], hazırladıkları makalede bir sorgunun; ilişkisel veri tabanları (RDBMS), ilişkisel olmayan (NoSQL) veri tabanları ve grafik veri tabanlarında nasıl olduğunu karşılaştırmıştır. Yaptığı uygulamada çoklu tablo sorguları, veri boyutu arttıkça ilişkisel veri tabanlarında etkin olmadığı sonucuna varmıştır.

Uzunbayır [13], yayınlanan çalışmasında ilişkisel ve grafik tabanlı iki farklı veri tabanı karşılaştırılmıştır. Bu iki teknoloji için veri modelleri tasarlanmıştır. Sorgu performans sonuçları gösterilerek grafik tabanlı tasarımlar için bilgi bankası çıkarılmıştır.

Aladily [14], hazırladığı yüksek lisans tezinde 4 ayrı NoSQL veri tabanının performansı karşılaştırılmıştır. Çalışma; Couchdb, Mongodb, Cassandra ve Hbase mimarisi ile ilgili bilgiler ile bu veri tabanlarının kurulumunu ve analizini içermektedir.

Karataş [15], yüksek lisans tezinde yazılım mühendisliği kapsamında önemli bir konu olan yazılım testlerinin, kullanımı artan dağıtık sistemlere uygulanmasını amaçlanmıştır. Bir dağıtık veri tabanı uygulaması olan MongoDB üzerinde farklı sınamaların gerçekleştiği test uygulaması geliştirilmiştir. Bu uygulama kullanılarak dağıtık veri tabanları üzerinde yapılan iyileştirmelere ait sonuçlar paylaşmıştır.

Ayyagari [16], hazırladığı yüksek lisans tezi ile arabuluculuk sistemleri için uygun veri tabanlarını sorgulamıştır. Ölçeklenebilirliğin performansa etkileri üzerine araştırma yapmıştır.

Aydemir [17], yayınlanan çalışmasında sınır güvenliği için gerçek zamanlı akan veri ve yığın veri üzerinde analiz yapmayı sağlayan tutarlı, dayanıklı, dağıtık ve ölçeklendirilebilir bir sistem örneği geliştirmiştir. Geliştirilen sistemin test edilebilmesi için örnek senaryolar oluşturmuş ve elde edilen sonuçları gözlemlemiştir.

Hammood [18], hazırladığı tezinde NoSQL veri tabanı sistemlerinden yaygın olarak kullanılanları seçerek, yeteneklerini ve farklı operasyonlarda nasıl tepki verdiklerini ortaya çıkarmak için detaylı testler yapılmıştır.

Taha [19], çalışmasında veri tabanı sistemleri ile ilgili detaylı bilgi vermeyi müteakip, “Yahoo Cloud Serving Benchmark” uygulamasını kullanarak; veri ekleme, okuma, güncelleştirme ve silme gibi temel işlemler bakımından MySQL, Apache, Cassandra ve MongoDB veri tabanlarının performans karşılaştırmasını yapmıştır.

Eren [20], hazırladığı tez çalışmasında bulut bilişim teknolojileri ile NoSQL veri tabanlarını incelenmiş ve hazırlanan uygulama ile bu teknolojilerin ne kadar efektif olduğunu araştırmaya çalışmıştır.

Literatürde yer alan kaynaklarda; farklı veri tabanı sorguları kapsamında kullanıcılara hangi uygulamalarda hangi veri tabanını seçmeleri konusunda bir öneri veren çalışmaya rastlanmamıştır. Aynı zamanda büyük veri ile başa çıkabilen NoSQL veri tabanı teknolojilerin yönetim bilişim sistemlerinde yaygın olarak kullanılan uygulamalar kapsamında ele alındığı bir çalışmaya da rastlanmamıştır. Oysa ilişkisel ve NoSQL veri tabanı özelliklerini iyi bilerek, hazırlanan uygulamaların niteliklerine en uygun veri tabanı seçmek gerek maliyet gerekse de hız açısından oldukça önemlidir.

Bu kapsamda yapılan çalışmada; öncelikle kurum/firmaların kullanabileceği alternatif veri tabanları açıklanmış ve NoSQL veri tabanlarının dört veri modeli tek tek ele alınarak işletmeler açısından güçlü ve zayıf tarafları ortaya konmuştur. Mevcut durumda firmaların kullandığı veri tabanı sistemleri ile ilgili araştırma sonuçlarına da yer verilerek, ilişkisel ve NoSQL veri tabanlarının avantaj ve dezavantajları; yatay ve dikey ölçeklenebilirlik, ACID ve BASE kuralları, tam ve yarı veri yapıları ile yazılan ve okunan şema parametrelerine göre ele alınmıştır. Firmaların yönetim bilgi sistemlerince çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanlarının farklı veri tabanı sorguları karşısında gösterdikleri reaksiyonlar karşılaştırılmıştır. Elde edilen test sonuçları analiz edilerek, yeni geliştirilecek uygulamalarda kullanıcılara secim konusunda fikir verilmiştir. Firma veya kurumların yönetim bilgi sistemlerince geliştirilmiş olan mevcut uygulamalarda köklü değişiklikler yaparak, NoSQL veri tabanı yönetim sistemlerinin sağladığı bilinen hız avantajını elde etmelerinin, işletme verimliliği üzerindeki etkileri sorgulanmıştır.

3. BİLİŞİM SİSTEMLERİ VE VERİ TABANI TEKNOLOJİLERİ

Bilgi, günümüzde stratejik bir kaynak olarak değerlendirilmektedir. Bu durum bilginin yönetimini önemli kılmaktadır. Bilgi sistemleri sürekli olarak bilginin üretilmesi, yenilenmesi ve yayılmasını sağlayarak işletmelerin yönetim fonksiyonlarına ve iş süreçlerine katkıda bulunmaktadır. Bilgi sistemleri kapsamında geliştirilen uygulamalarda veri tabanı yönetim sistemlerinin rapor üreteçlerinden yararlanılarak karşılaşılabilecek problemlerinin çözümüne olanak sağlanmaktadır. Yapılan veri tabanı sorgulamaları ile toplanan veriler, rapor üreteçleri vasıtası ile belirli şablonlara göre düzenlenerek yöneticilerin karar vermelerini desteklemek üzere raporlanabilmektedir.

3.1. Bilişim Sistemleri, Faaliyetleri ve Yaklaşımları

Bilişim sistemi, organizasyonlarda karar verilinceye dek bilgi toplamak, toplanan bilgiyi düzenleyerek işlemek ve saklamak olarak tanımlanmaktadır. Bilişim ise; verilerin kullanıcı için faydalı ve anlamlı hale getirilmiş biçimi olarak tanımlanmaktadır. Kontrol ve koordinasyon sağlayan bilişim sistemleri; kullanıcılar tarafından karmaşık problemlerin analiz edilerek çözülmesine yardım etmekte ve kullanıcıların karar vermelerine destek olmaktadır. Bilişim sisteminde bilgi üretimi için gerekli üç aktivite vardır. Bunlar; girdiler, işlemler ve çıktılardır. Girdiler, organizasyon içinde veya organizasyon dışında bulunan ve henüz işlenmemiş olan verileri toplamaktır. İşlemler, henüz ham durumda olan verilerin anlamlandırarak faydalı hale getirilmesidir. Çıktılar ise, işleme tabi tutulmuş bu bilgileri kullanıcılara ve ilgili aktivitelere ulaştırılmasıdır [21].

Bilişim sistemlerinin temel bileşenleri

Bilişim sistemleri işletmeler açısından değerlendirildiğinde, ham verilerin işlemlere sokularak çıktıların elde edildiği mekanizmalardan daha geniş bir anlam taşımaktadır. Bilişim sistemleri çok ciddi teknolojik alt yapılardan yararlanan yönetim ve organizasyon çözümleri ihtiva etmektedir. Bilişim sistemleri etkin olarak kullanılmak isteniyorsa mutlaka, teknoloji, yönetim ve organizasyona hâkim olunması gerekmektedir [22].

Şekil 3.1'de bilişim sistemleri bileşenlerinin genel gösterimi verilmiştir.



Şekil 3.1. Bilişim sistemleri bileşenleri [22]

Bilişim sistemlerinin sınıflandırılması

Bilişim sistemleri organizasyon seviyelerine göre dört şekilde sınıflandırılabilmektedir:

I. *İşlem seviyesi bilişim sistemleri:* Bu sistemler, bir organizasyona ait satış bilgileri veya bir fabrikaya ait malzeme akış bilgileri gibi temel aktivite ve işleri kaydederek işlem seviyesindeki yöneticiyi desteklerler. Temel prensip rutin olan soruları cevaplamak ve kayıt tutulmasını sağlamaktır. İşlem seviyesi sistemlerine örnek olarak, bankamatik veya çalışılan süreyi hesaplayan sistem gösterilebilir.
II. *Bilgi seviyesi bilişim sistemleri:* Bu sistemler, bir organizasyona ait bilgilerin desteklenmesini sağlar. Firmaya ait ofis işlemlerinin kontrol edilmesine ve yeni elde edilen bilginin iş ile bütünleşmesine olanak sağlar. Bu sistemler halihazırda diğer seviyelere oranla daha hızlı gelişme göstermektedir.
III. *Yönetim seviyesi bilişim sistemleri:* Bu seviyedeki sistemler, orta düzey yöneticinin gelişmeleri izleyebilmesine, kontrol edebilmesine ve karar verebilmesine imkân sağlar. Bu sistemler anlık olarak hazırlanan raporlardan ziyade dönemsel olarak bilgi toplarlar. Periyodik rapor düzenlerler.
IV. *Stratejik seviye bilişim sistemleri:* Bu seviyedeki sistemler, üst düzey yöneticinin organizasyon içindeki ve dışındaki stratejiyi belirlemelerine yardımcı olmaktadır. Temel prensip dış çevredeki gelişmelere mümkün olan firma olanakları ile uyum sağlayabilmektir [22].

Bilişim sistemleri faaliyetleri

Bilişim sistemleri faaliyetleri; verilerin toplanması, işlenmesi, depolanması ve bilgisayar ağları vasıtası ile istenilen yerlere güvenle iletilmesinden oluşmaktadır. Bilgisayarlar vasıtası ile tasarım, işletme ve yönetim desteği sunan bu sistemler; yazılım, donanım ve iletişim teknolojisini bütünleştirmektedir. Bilişim sistemi, bilgilerin işletilip paylaşıldığı her ortamda kullanılmaktadır. Kullanıcılar, bu bilişim sistemlerini ihtiyaca göre yapılandırmaktadır [23].

Bilişim sistemleri yaklaşımları

Teknolojinin hızlı gelişmesi, bilgi erişim hızındaki artış, küreselleşen dünya ve rekabetin her geçen gün artması bilgiyi rekabetin temel unsuru haline getirerek, bilgiye en kısa sürede ulaşım sağlanmasını ve elde edilen bilginin etkinliğinin arttırılmasını zorunlu kılmıştır. Bu durum, bilgi sistemlerindeki işlevsel anlayışın yanı sıra modern bilgi sistem yaklaşımını beraberinde getirmektedir [24]. Bilgi sistemi yaklaşımları üç başlık altında incelenebilir.

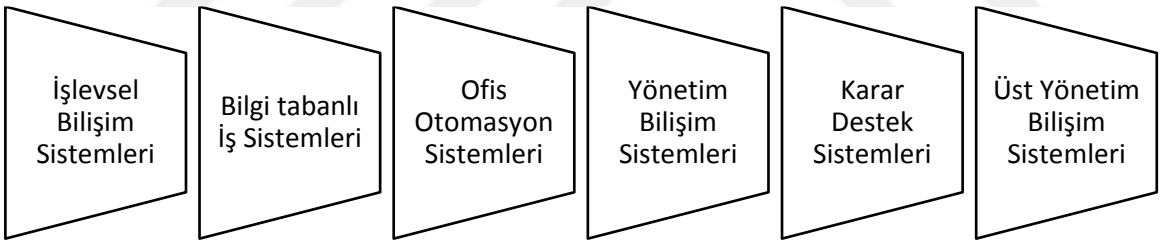
Teknik yaklaşım: Bu yaklaşım, bilgi sitemleri incelemesinde matematiksel temelli modelin önemli olduğunu vurgular. Bilgisayarlar, yönetimler ve yöneylem araştırması bu yaklaşıma katkı sağlamaktadır. Bilgisayar bilimleri; hesaplama, veri depolama ve ulaştırma yöntemleri ile ilgilenirken yönetim bilimleri; karar verme ve yönetimsel uygulamalar için model geliştirmeye odaklanır. Yöneylem araştırması ise stok kontrolü, taşıma ve üretim maliyeti gibi organizasyonun seçili parametrelerinin optimize edilmesi için matematiksel tekniklerin üzerinde durmaktadır [24,25].

Davranışsal yaklaşım: Bilgi sistemlerinde ortaya çıkan davranışsal konular ile ilgilenmektedir. Stratejik organizasyonların birleşmeleri, yönetimi ve tasarımı gibi hususlar, teknik yaklaşımlar ile incelenmesi zor konulardır. Davranışsal yaklaşımda yer alan modeller bilişim sistemlerine önemli katkı sağlar. Bu yaklaşımla; sosyoloji bilimi, ekip veya takımın sistemi nasıl geliştirdiği ve birey, grup ve tüm örgütün nasıl etkilendikleri üzerine incelemelerde bulunulabilmektedir [24,25].

Sosyoteknik yaklaşım: Yönetim bilişim sistemi hakkındaki çalışma ve araştırmalar, yönetici grupların bilgisayarlara yönlendiği 1970'li yıllara denk gelmektedir. Bu sistemler; yönetim bilimi, bilgisayar bilimi ve yöneylem araştırmalarına dair çalışmaların birleştirilmesini sağlamıştır. Aynı zamanda sosyolojik, psikolojik ve ekonomik konular ile iç içedir. Sosyoteknik yaklaşım, bilişim sistemlerine yalnızca teknik açısından bakılması yanılgısını ortadan kaldırmaya yardımcı olur. Bu yaklaşımda davranışsal ve teknik hususlar beraber ele alınır. Teknoloji, bireysel ve örgütsel gereksinimlere binaen tasarlanır ve geliştirilir [24,25].

Temel bilişim sistemleri

Organizasyonlarda çeşitli seviyelerde ihtiyaç duyulabilecek bilgi gereksinimini karşılayabilecek farklı bilgi sistemleri mevcuttur. Bilgi teknolojileri gelişimi ile paralel olarak gelişen bu sistemleri altı ana grupta toplamak mümkündür [26]. Şekil 3.2'de temel bilişim sistemleri gösterilmektedir.



Şekil 3.2. Temel bilişim sistemleri [26]

1. İşlevsel bilişim sistemi: Günlük iş akış takibinin yapılabilmesi için, rutin kayıt tutulan sistemlerdir. İşlevsel bilişim sistemleri, günlük olarak kayıtların tutulması yanında ödeme, sipariş gibi dokümanlar için de kontrol sağlamaktadır. Satış ve sipariş sistemi, otel rezervasyonu sistemi veya çalışan bilgi bankası gibi sistemler işlevsel bilişim sistemlerine örnek olarak gösterilebilir. İşletmelerin işlevsel seviyesinde hizmet verir [26].

2. Bilgi tabanlı iş sistemi: Belirli bir alanda uzmanlaşan personele hitap eden, bilgi toplayıp bu bilgiyi iş akışlarına entegre eden sistemlerdir. Örnek olarak; bilgisayar destekli üretim ve tasarım ile sanal gerçeklik sistemleri gösterilebilir [26].

3. Ofis otomasyon sistemi: Ofis ortamlarında koordineyi ve iletişimsel aktiviteleri destekleyerek üretkenliği arttıran uygulamalara denir. Örnek olarak; kelime işlemcileri, doküman yönetimi, planlamayı sağlayan programlar ve iletişim sistemleri gösterilebilir [25].

4. Yönetim bilişim sistemi (YBS): Bilgi sağlanması, bilginin işlenmesi ve karar alınmasını kolaylaştıran sistemlerdir [27]. Yönetim bilişim sistemleri (YBS), bilgisayar sistemlerinin yönetimsel süreçlerde kullanılmaya başlaması ile gün yüzüne çıkın bir kavram olmuştur. Bilgi sistemlerinde bilgi, her ne kadar bilgisayarlar ve tamamlayıcı araçlarla işleniyor olsa da diğer organizasyonlarda basit yöntemler ile gerçekleştirilmektedir [28]. Yönetim bilgi sistemleri; çalışan grupların çabaları ile belirlenen hedefe ulaşmayı ifade eden "yönetim", gruplar arasında iletilen "bilgi" ve belirlenen bir hedefe uygun olarak hareket edebilen unsurlar olan "sistem" kelimelerinden oluşan bir kavramdır [29].

Yönetim bilişim sistemleri (YBS), bir işletmenin bütün kademelerinde bulunan yöneticilerin, sorumluluğundaki planlama, denetleme ve yönetimle ilgili aktiviteler için etkili ve verimli kararlar alabilmelerini sağlama amaçlı, organizasyonların içinden ve dışından elde edilen veriyi enformasyon haline getiren ve bunun iletilmesini sağlayan sistemlerdir. Yönetim bilişim sistemleri temelinde yatmakta olan bu enformasyon, bütün yönleriyle bilgi merkezlerini kapsamaktadır [30].

Günümüzde organizasyonların içinde ve dışında oluşabilecek bütün gelişmelere hızlı şekilde reaksiyon göstermek ve ilgili bilgiye anında ulaşabilmek kurumların geleceği açısından çok önemlidir. Bu nedenle kurumlarda bulunan karar vericileri, iç ve dış her türlü bilgi ile anlık olarak buluşturabilecek sistemlere ihtiyaç duyulmaktadır. Yönetim bilişim sistemleri, organizasyonların güncel ve tarihsel verilerine aynı anda ulaşarak raporlar vasıtası ile yöneticilere destek sağlamaktadır [31].

Yönetim bilişim sistemlerinin temel amacı yönetici kademelerinin bilgi ihtiyaçlarına cevap vermektir. Bu sistemler özellikle denetleme, planlama ve düzeltme faaliyetlerinde bulunulması maksadıyla geliştirilmiştir. İşletmelerin üretim, muhasebe ve finans gibi işlevlerine dair bilgiyi farklı araçlar vasıtası ile yönetici kademelere ileten sistemlerdir [32].

Yönetim bilişim sisteminin gelişimi: Gelişen bilgi sistemleri ve teknolojilerin, kurumsal ve toplumsal örgütlerin verimliliğine son derece katkı sağlayabildiği; buna karşı olarak ise söz konusu örgütlerde bulunan hiyerarşinin bozulmasına yol açtığı belirtilmektedir. Sebep olarak ise kullanımı sağlanan bu sistemlerin; önceden hiyerarşik olarak iletilen bilginin artık direk olarak ilgili kişiye ulaştırılmasının sağlanması gösterilmektedir [22].

Yönetim bilişim sistemlerinin sınıflandırılması: Bu sistemleri çeşitli şekillerde sınıflandırılmak mümkündür. Mesela bu sistemler, operasyonel destek, yönetim destek ve diğer sistemler olarak üç başlık altında sınıflandırılabilir. Bu durumda; süreç kontrol, kurumsal iş birliği ve veri işleme sistemlerini operasyonel destek sistemleri, yönetici bilgi, karar destek ve yönetim bilişim sistemlerini yönetim destek sistemleri ve bilgi yönetim, stratejik bilgi, işlevsel işletme ile uzman sistemleri ise diğer bilgi sistemleri olarak düşünmek mümkündür [33].

Yönetim bilişim sisteminin özellikleri: Yönetim bilişim sistemleri (YBS) yönetici kademesindeki kişilerin verilerin analiz edilerek karar vermelerini desteklemek için oluşturulmuş sistemlerdir. Bu sistemlerin kaliteli olabilmesi için yöneticilere ihtiyaç duydukları bilgiyi tam ve zamanında sağlayabilmeleri gerekmektedir. Bilgiler kesinlik ve güncellik ihtiva etmelidir. Kullanıcılar farklı bakış açıları ve gereksinimlere sahip oldukları için bu bilgilere; istatistik, grafik veya tablo gibi farklı formatlarda ulaşabiliyor olmalarını sağlayacak esneklikte olmalıdır [34].

Laudon'a [35] göre yönetim bilişim sistemlerinin temel özellikleri şöyle sıralanmaktadır:

- Kontrol düzeyinde yapılandırılmış ve yarı yapılandırılmış kararları destekler.
- Üst yönetimin planlama amaçlarına katkıda bulunur.
- Genellikle raporlamaya dönük ve kontrol odaklıdır.
- Mevcut şirket verileri ve bu verilerin akışına dayanır.
- YBS'nin analiz yeteneği düşüktür.
- Karar verme sürecine geçmişteki ve mevcut veriler yardımıyla katkıda bulunur.

- YBS daha çok örgüt odaklıdır. İhtiyaç duyulan bilgi genellikle sabittir ve bilinmektedir. Uzun bir analiz ve tasarım sürecini gerektirmektedir.
- Veri / kayıt işleme fonksiyonlarını destekler.
- Bütünleşik bir veri tabanı kullanır ve fonksiyonel alanların çeşitliliğini destekler.
- Operasyonel, taktik ve stratejik seviye yöneticilerin bilgiye kolay ve zamanında erişimini sağlar.
- Kısmen esnektir ve organizasyonun bilgi ihtiyaçlarındaki değişmeye adapte edilebilir.
- Sadece yetkili şahısların erişimine imkân veren sistem güvenliği sağlar.
- Günlük operasyonlarla ilgilenmez.
- Genellikle yapısal kararların desteklenmesine yöneliktir.
- Yöneticilere değişik raporlar sunar.
- Öncelikle çevresel ya da dış olaylara değil, büyük ölçüde firma içi olaylara odaklanır.

Yönetim bilişim sisteminin faydaları: Bu sistemler, organizasyonlar tarafından karar alma ve örgütsel süreçlerin yönetimi maksatları ile kullanıldığında şu faydaları sağlamaktadır [36];

- Cari işlemlerin daha ucuz ve daha hızlı yapılması ile operasyonel verimliliği arttırır.
- Özellikle turizm ve bankacılık gibi sektörlerde işlemlerin kısalmasına ve kullanıcılara daha iyi bir hizmet sunulabilmesine imkân sağlar.
- Bilgi günümüzde değerli bir üretim girdisi olması yanı sıra özellikle hizmet sektöründe bizatihi ürünün kendisini olmuştur. Bu sistemler bilgi temelli yeni ürünlerin oluşturulmasına destek olmaktadır.
- Bilgiyi rakiplerine göre daha iyi işleyebilen işletmeler rekabet üstünlüğü elde etmektedir. Bu kapsamda ciddi rekabet avantajı sağlanmaktadır.
- Bu sistemler sayesinde fırsatların fark edilmesi ve yakalanması kolaylaşır.
- Organizasyonların iletişim imkanlarını arttırarak, çalışanlar ile üst yönetimin koordinasyonunu sağlar ve stratejik planlamalara destek olur.
- Kurumsallaşmaya olan katkısı yanı sıra zaman ve emek tasarrufu sağlar.

Gökçen [37]' de yönetim bilişim sistemlerinin faydalarını şu şekilde sıralamaktadır;

- İşin büyümesine kolaylık sağlar.
- Ofis personelini azaltır.
- Bilgi işleme maliyetlerini azaltır.
- Yapısal kararların otomasyonunu sağlar.
- Daha fazla ve daha iyi bilgi sağlar.

Polat [38] ise yönetim bilişim sistemlerinin kazanımlarını şu şekilde sıralamaktadır;

- Bilgi akışında eşgüdüm ve standart sağlanır.
- Tüm iş süreçlerine ait bilgilerin sistematik bir şekilde toplanması ve bir araya getirilmesi sağlanır.
- Personel ve zaman açısından önemli kazanımlar elde edilir.
- Bilgiye güven duyulması ve oto-kontrol mekanizmalarının kurulması sağlanır.
- Bilginin gizlenme veya kaybolma riskinin ortadan kalkması sağlanır.
- Yöneticilerin, is süreçlerine ve kurumun işleyişine hâkim olmasını kolaylaştırır.
- Yöneticilerin hızlı, güvenilir ve bilgiye dayalı olarak karar alabilmesini sağlar.

5. Karar destek sistemi: İleri seviyede, rahatlıkla tanımlanabilir olmayan, hızlıca değişiklik gösteren kararların verilmesinde yöneticilere destek veren sistemlerdir [25]. Karar destek sistemleri; bilgi teknolojileri, yazılım ve insan zekasının, karşılaşılan zor problemlerin çözümü için en etkili şekilde bir araya getirilmesidir [39]. Karar destek sistemleri sahip olduğu düzeltme, raporlama ve kolay kullanım fonksiyonları ile [40], yönetici seviyesinde karar alma desteği sağlamak için tasarlanan bilgi sistemleridir [41]. Bu sistemler; İşlevsel bilişim sistemi ve yönetim bilişim sisteminden organizasyon içi bilgi almakla birlikte organizasyon dışı rakip firma bilgilerini de kullanmaktadır. Tasarlanış olarak diğer bilgi sistemlerinden analitik güç bakımından daha üstün olan bu sistemler, büyük miktarlarda özet verileri raporlayabilmektedir. Bu kapsamda mümkün olduğunca değişen durumlara ve problemlere gerekli çözümü üretebilecek şekilde esnek olarak tasarlanması gerekmektedir [31].

6. Üst yönetim bilişim sistemi: Stratejik seviyede organizasyonlarca kullanılan sistemlerdir. Organizasyonların misyon, vizyon, değer ve stratejilerinin belirlenmesinde rol alan üst seviyeli yöneticilere hitap edecek şekilde oluşturulmuş sistemlerdir. Bu seviyede yapılabilecek hata organizasyon açısından büyük risk teşkil edeceği için bu sistemlerin iç ve dış çevre koşulları hususunda tam ve doğru şekilde bilgilendirilmesi sağlanmalıdır [25]. Bu kapsamda üst yönetim bilgi sistemleri; önceden programlanması zor, karmaşık ve stratejik kararların verilmesinde kullanılan bilgi sistemleri olarak tanımlanabilmektedir. Bu sistemler ile, grafikli gösterimle desteklenen ve çok çeşitli kaynaklardan elde edilen veriler üst yöneticilerin hizmetine sunulur. Kritik olarak nitelenen bilgiye ulaşım için harcanan zamanın, mümkün olduğunca en aza indirilmesi temel hedeftir [24].

3.2. Veri Tabanı Tanımı, Tasarımı ve Optimal Performansı

Tekil hali Latincede “datum” olan kelimeden türetilen verinin, sözcükteki anlamı gerçektir. Ancak veri her durumda gerçekleri göstermemektedir. Kesin olmamakla birlikte bir fikrin tarifi için de kullanılabilir. Kullanım şekline bakıldığında veri, kullanıcılar tarafından kaydetmeye değer bulunan olaylar ve fikirler olarak ifade edilmektedir. Veri; olgu, kavram ve talimatların insan tarafından veya otomatik yolla işleme, yorumlama ve iletişim amaçlarına uygun şekilde ifadesidir [42].

Veri tabanı tanımı

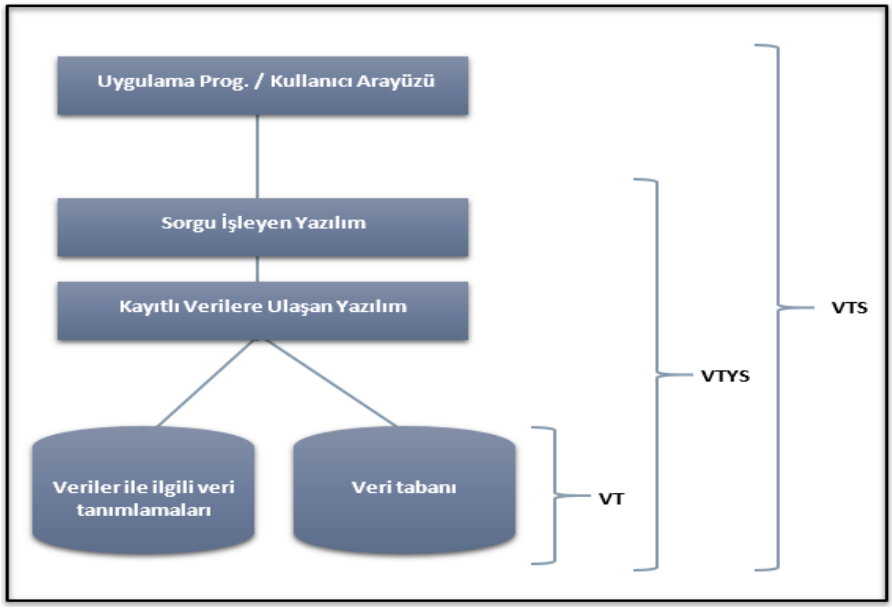
Veri tabanı; genel olarak, kullanıcıların amaçlarına uygun düzenlenen veri toplulukları olarak tanımlanmaktadır. Birbiri ile ilişkisi olan verilerin bir arada tutulduğu, fiziki ve mantıki tanımlamalarının var olduğu bilgi deposudur. Veri tabanları, var olup birbirleri ile ilişki içinde bulunan nesnelerin ve ilişkilerin modellenmesidir [42]. Veri tabanları; hastane, üniversite gibi çeşitli organizasyonların çalışarak işlediği farklı uygulama programlarınca kullanılan verilerin toplamıdır. Verilere örnek olarak; bir şirketin müşteri bilgileri, satış, ürün veya ödeme bilgileri, bir hastanenin doktor, hasta, teşhis bilgileri gibi çok çeşitli bilgiler gösterilebilir. Veri tabanı ile veriler kaydedilebilir, silinebilir, güncellenebilir, yeni veriler eklenip mevcut veriler üzerinde sorgulama yapılabilmektedir.

Veri tabanı yönetim sistemi (VTYS)

Veri tabanı yönetim sistemi; veriler üzerinde ihtiyaç duyulan standartları oluşturan mantığı sağlayan sistemlerdir [42]. Veri tabanı yönetim sistemleri, veri bütünlüğü güvenliği ve güvenilirliği gibi modern veri tabanlarının standart özelliklerini sağlamalıdır. Veri tabanı yönetim sistemleri, veri tabanı içinde depolanan verilere erişim sağlar. Bu sistemler verilere aynı anda birden çok bağlantı sağlayabilme özelliğine sahiptir. Veri tabanı yönetim sistemi, veri tabanı yönetiminin bir parçası olarak, verinin nasıl depolanacağı, kullanılacağı ve erişileceğine mantıksal olarak yön veren kurallar sistemi olarak tanımlanabilmektedir. Veri tabanı yönetim sistemi veri tabanlarını da içermektedir.

Veri tabanı sistemi

Veri tabanı, veri tabanı yönetim sistemi ve bunlara ek olarak uygulama programları ile kullanıcı ara yüzlerini içeren tümleşik yapıya veri tabanı sistemi denir. Veri tabanı sistemi; veri tabanlarının kurulmasını, oluşturulmasını, tanımlanmasını, işletilmesini ve kullanılmasını sağlayan programlar topluluğu olarak ifade edilmektedir [43]. Şekil 3.3'te veri tabanı, veri tabanı yönetim sistemi ve veri tabanı sistemi arasındaki ilişki ve işlevler gösterilmiştir.



Şekil 3.3. VT, VTYS ve VTS arasındaki ilişki ve işlevler [42]

Veri tabanı modelleri

1. Düz veri modeli: Bu model iki boyuttan oluşan veri grubu olarak ifade edilir. Bu boyutların parçalarından biri olan sütunda, benzer veri özellikleri bulunurken diğer parçaları oluşturan satırlarda veri grupları yer almaktadır. Örnek olarak, kullanıcı adı ve şifrelerin tutulduğu güvenlik veri tabanları gösterilebilir. Bu örnekteki gibi bir veri tabanında satırlarda kullanıcı şifreleri tutulurken, aynı tipteki veriler sütunlarda tutulabilmektedir. Bu modeli tek tablodan oluşan model olarak düşünmek mümkündür [44]. Çizelge 3.1'de düz veri modeli örneği verilmiştir.

Çizelge 3.1. Düz veri modeli örneği [44]

	Adı Soyadı	Kullanıcı Adı	Parola
Kayıt 1	Murat ERGİN	Mergin	kjVdb125
Kayıt 2	Ayşe YILMAZ	Ayılmaz	Bks46db7
Kayıt 3	Can TÜRK	Cturk	fhG8dbt9

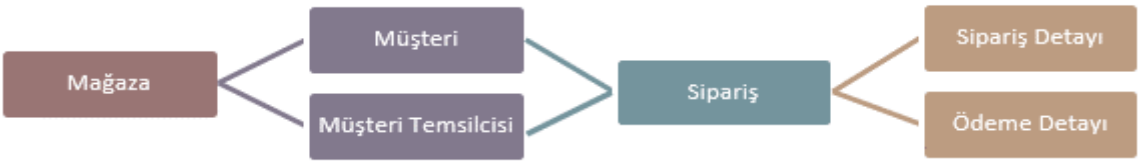
2. Hiyerarşik veri modeli: Bilgi sistemleri, hiyerarşik veri tabanı modeli sayesinde veri tabanı sistemlerinde dosya depolama yönünde gelişim göstermiştir. İlk olarak 1960'lı yıllarda ortaya çıkmış ve adını veriyi depolama yönteminden almıştır. Bu veri tabanının depoladığı yapısal verilere kayıt adı verilmektedir. Kayıtlar ağaç mimarisi şeklinde yukarıdan aşağı doğru sıralanmaktadır. Kök adı verilen ilk kaydın bir ya da daha fazla çocuk kayıtları vardır. Çocuk kayıtlarında kendi çocuk kayıtları olabilmektedir. Kök haricinde bütün kayıtların bir ebeveyni vardır. Kök kayıt "hiyerarşiyi" şekillendirdiğinden, kendisine ait bir ebeveyni yoktur. Veri kayıtları arasındaki ilk ilişki hiyerarşik veri tabanları ile ortaya çıkmıştır [45]. Burada kayıtlar arasında birden çoğa ilişki kurulabilmiştir. Böylece her müşterinin, satın aldığı ürün sayısı kadar değil, sadece bir kere kaydedilmesi sağlanmıştır. Şekil 3.4'te hiyerarşik veri tabanı modeli örneği verilmiştir.



Şekil 3.4. Hiyerarşik veri tabanı modeli [45]

Modern veri tabanları ile karşılaştırıldığında hiyerarşik veri tabanı modelinin kendine özgü kısıtlamaları vardır. Bunlardan biri, her veri tabanı içinde sadece bir ağaç yapısının olabilmesidir. Şekil 3.2'de gösterildiği gibi; "bir öğrenci bir ya da daha fazla ders alabilir" ve "bir öğretim üyesi bir ya da daha fazla ders verebilir". Birden fazla ebeveyn tablosu kullanmak mümkün olmadığından, okuldaki ders kayıtlarının tekrarlanması gerekir. Ayrıca eğer bir derste değişiklik olursa, sınıf kayıtlarının da güncellenmesi gerekir. Bu nedenle veri tekrarı hala önemli bir sorundur [45]. Diğer bir kısıtlama ise, veri tabanı kayıtları arasında dolaşmaktır. Kullanıcının herhangi bir kayda erişmesi için ağacın kökünden başlaması, istediği kayda erişene kadar ebeveynden çocuğa doğru ilerlemesi gerekir. Bu nedenle kullanıcılar verinin depolanma şeklini bilmelidir.

3. Ağ veri modeli: Hiyerarşik veri modelinin gelişmiş bir hali olan bu model, yetmişli yıların başlarında ortaya çıkmıştır. Verinin doğal olarak çok sayıda veriyle ilişkili olabilmesinin sonucu olarak hızlı bir biçimde kabul görebilmiştir. Ağ veri modelinde verilerin birbirlerine ağ biçiminde bağlı oldukları varsayılır. Hiyerarşik modelle olan en büyük fark, uç-düğüm pozisyonundaki verilerin iç-düğüme işaret edebiliyor olmasıdır. Bu sayede bu model vasıtası ile bire-çok ilişki yanı sıra, çoka-çok ilişki de modellenebilmektedir. Bu durum verinin tekrar etmesini ciddi oranda azaltmaktadır [46]. Şekil 3.5'te ağ veri modeli örneği verilmiştir.



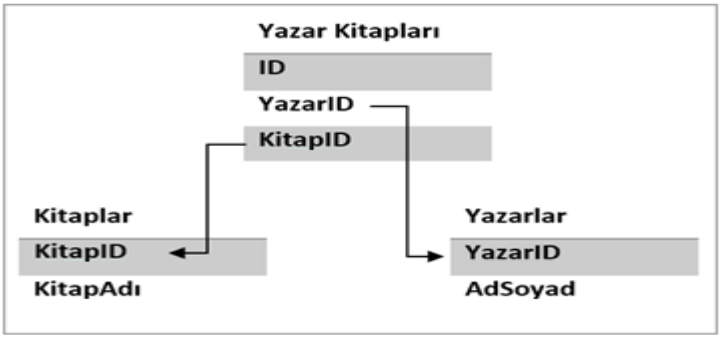
Şekil 3.5. Ağ veri modeli [46]

4. İlişkisel veri modeli: Ağ ve hiyerarşik modellerin her geçen gün artan ihtiyaçlara cevap verememesi, gelişmiş bir model ihtiyacını doğurmuş ve yapılan çalışma ve araştırmalar neticesinde ilişkisel veri modelleri ortaya çıkarılmıştır. E. F. Codd'un 1970'de yazmış olduğu "A Relational Model of Data for Large Shared Data Banks" adlı makale ilişkisel veri yapılarında çığır açmıştır. Bu önemli çalışmada, Codd ilişkisel mantığı tanıtarak tablolar vasıtası ile gerçek nesnelerin nasıl gösterilebileceğini belirtmiştir [47].

Bu modelde temel kavram ilişkidir ve çeşitli ilişki örnekleri, modeli meydana getirir. İlişkiler sütun ve satırlardan oluşan iki boyutlu tablolar vasıtası ile belirtilir. Veri tabanının sütunlarında nitelikler bulunurken satırlarda birbirleri ile ilişki içindeki veriler yer almaktadır [48].

İlişkisel veri modellerinde, tabloların içindeki satır sıralarının bir önemi yoktur ancak iki farklı satır birbirlerinin aynısı olmamalıdır. Tablolardaki sütunların sayısı sabitken, satırlar için herhangi bir sınırlama yoktur. Farklı tablolar arasındaki bağlantı anahtar olarak adlandırılan sütun değerleri vasıtası ile sağlanmaktadır. Anahtar değerler sayesinde kullanıcı ihtiyaç duyduğu veriyi farklı tablolardan sorgulayarak alabilmektedir.

İlişkisel veri tabanı modellerini diğer modellerden ayıran en önemli fark, bu modelde ilişkilerin tamamının veri tabanlarının oluşturulması esnasında tanımlanması zorunlu değildir. Temel ilişkilerin oluşturulması sonrası istenildiği zaman ihtiyaç duyulan yeni ilişkiler türetilebilmektedir. Bu sayede kullanıcıların zamanla değişebilen isteklerine cevap verebilecek esneklik sağlanmış olmaktadır. Şekil 3.6'da bir ilişkisel veri modeli örneği gösterilmiştir.



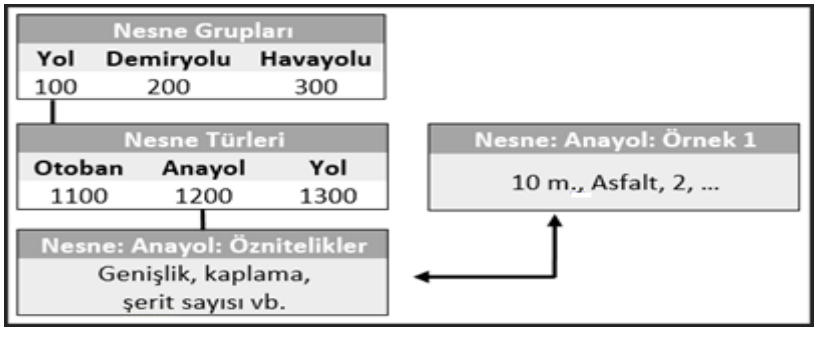
Şekil 3.6. İlişkisel veri modeli [48]

Ortaya çıkmaları ile hızlıca kullanılmaya başlanan bu modeller, günümüzde de yaygın olarak kullanılmaya devam etmektedir. MySQL, ORACLE, MSSQL gibi veri tabanları, ilişkisel veri tabanı modellerine örnek olarak gösterilebilir. Orijinalinde çok karmaşık olmayan veriler için geliştirilmesi nedeni ile konum gibi karmaşık veriler kullanan çeşitli uygulamalarda, yaşanan sıkıntılara karşı zamanla farklı ilişkisel veri tabanları geliştirilmiştir [46].

5. Nesne yönelimli veri modeli: İnternetin yaygınlaşmaya başladığı 1990'lı yılların ortasından itibaren ortaya çıkan bu model, nesne yönelimli programlamalara dayanmaktadır. Karmaşık uygulamalarda başarılı olduğunu kanıtlamıştır. Bu modelin tasarımında; gittikçe zorlaşan programlama problemleri ile başa çıkabilen nesne yönelimli modelin, verilerin modellenmesinde de etkili olabileceği varsayımından yola çıkılmıştır [46]. Bu veri modellerinin çekirdeğini oluşturan hususlar şunlardır:

- *Nesneler ve nesne tanımlayıcıları:* Bu sistemlerde gerçek hayatta bulunan varlıklar nesneler olarak modellenmektedir. Bu nesnelerin ayrıca tanımlayıcıları bulunmaktadır.
- *Öznitelik ve metotlar:* Nesnelerin durumları ve davranışları vardır. Durum nesnenin özniteliğinin almış olduğu değerler kümesidir. Davranış ise nesnelerin durumları üzerindeki metot kümesidir. Nesnelerin içine yerleştirilmiş durum ve davranışa, yalnızca dışarıdan gönderilebilen mesaj (fonksiyon çağrısı) vasıtasıyla ulaşılır.
- *Sınıflar:* Aynı metot ve öznitelikler kümelerini paylaşan nesneleri gruplandırma aracı olarak kullanılmaktadırlar.
- *Sınıf hiyerarşileri ve kalıtımı:* Bu sistemlerde, mevcut bir sınıftan yeni bir sınıf türetilebilmektedir. Bu yeni sınıfa alt sınıf denir. Bu sınıf; önceden var olan ve üst sınıf olarak adlandırılan sınıfın bütün özniteliklerini ve metotlarını kalıtsal olarak taşımaktadır. Kimi sistemlerde altsınıfların sadece bir üst sınıfı varken kimi sistemlerde birden fazla olmasına izin verilebilir. İlki tekil katılım olarak adlandırılırken diğerine çoklu katılım denilmektedir [46].

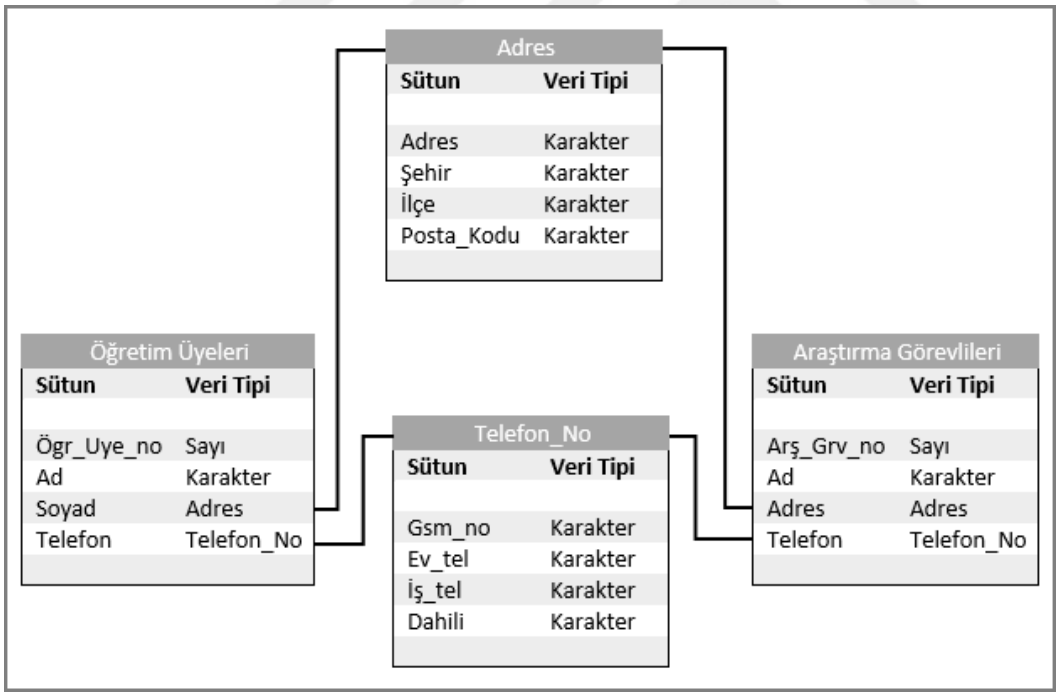
Şekil 3.7'de nesne yönelimli bir veri modeli örneği gösterilmiştir.



Şekil 3.7. Nesne yönelimli veri modeli [46]

Bu modelde, yapılan sorguların karşılığı olarak mutlaka önceden belirlenmiş bir nesne kümesi olmalıdır. İlişkisel veri tabanı modellerindeki ilişki kavramına karşılık olarak bu modellerde sınıf kavramı yer almaktadır. Önemli avantajlarından biri, ortak metot ve özniteliklere sahip nesnelerin farklı ortamlarda yeniden tanımlanmalarına gerek olmamasıdır. Bu sayede gereksiz yer kaybından kaçınılabildiği gibi dinamizm de sağlanmış olmaktadır. Bunun nedeni, ilgili sınıfın ait olduğu üst sınıfa dair bütün özellikleri taşımasıdır [49].

6. Nesne ilişkisel veri tabanı modeli: Bu model, ilişkisel özellikler yanı sıra nesne yönelimli özellikler de içermektedir. Bu veri modelinde amaç; ilişkisel tipteki ortamlarda nesnel olarak veri depolamaktır. Bir veri tabanı yönetim sistemi nesne ilişkisel olabilmesi için, kullanıcı tanımlamalı veri tipi, katılım ve nesne desteği sağlayabilmelidir. Şekil 3.8'de nesne ilişkisel veri modeli örneği gösterilmiştir.



Şekil 3.8. Nesne ilişkisel veri modeli [50]

Parmak izi analizi ve depolanması için genellikle bu tip veri tabanları kullanılmaktadır. Bu veri tabanları sayesinde öncede fotoğraflar ile ilişkilendirilen parmak izleri binlerce benzerleri arasından kısa süre içerisinde bulunabilmektedir [45].

7. Çoklu ortam veri modeli: Bu modeldeki veri tabanları nesne ilişkisel veri tabanı sistemleriyle benzerlik göstermektedir. Ancak bu sistemlerle; müzik, video ve film gibi büyük nesnelerin işlenmesi ve işlem adımlarının kullanıcılara gösterilmesi de mümkündür. Çoklu ortam veri tabanlarının desteklemesi gereken üç temel özellik; veri miktarı, süreklilik ve senkronizasyondur [45]. Bu veri modelleri yüksek miktarlarda depolama, bellek ve ağ donanımı kapasitesi ihtiyacı duymaktadır. Ağ üzerinden veri aktarım hızı oldukça yüksek olmak durumundadır. Bu modellerdeki veri tabanı sistemleri daha çok tıp bilgi sistemlerinde kullanılmaktadır. Örnek olarak üç boyutlu tıbbi görüntü kaydı, uzaktan görüntülü eğitimler ve imge görüntüleme verilebilir [45].

8. Dağıtık veri modeli: En az iki bilgisayar üzerinde depolanıp bir ağ üzerinden dağıtımı sağlanan bilgiler için kullanılmakta olan veri tabanı modelidir. Dağıtık veri tabanlarında bilgisayar ağları üzerinden paralel kullanım için parçalara ayrılma, sorguların hızlı şekilde işlenmesine olanak sağlamaktadır. Bu model, kullanıcılara tek bir veri tabanına erişim sağlandığını hissettirmesine rağmen aslında birden fazla veri tabanına erişim söz konusudur. Bu yeni veri tabanı modellerinin henüz çözmesi gereken bazı sorunları mevcuttur [45]. Bu sorunlar şu şekilde sıralanabilmektedir:

- *Dağıtık sorgularını işlenmesi:* Sorgu, bilgilerin okunması için ağ üzerinde parçalara ayrılırken, sonucun doğru bir biçimde bir araya getirilmesi gerekmektedir.
- *Dağıtık eşzamanlılık kontrolü:* Birden fazla veri tabanlarının birbirleri ile uyumlu bir şekilde çalışması çözülmesi zorunlu olan problemlerden biridir. Örnek olarak; bir veri tabanında işlem yapılırken bu işlem sonuçlanmadan bilgiler üzerinde güncellemeye izin verilmemelidir.
- *Dağıtık çıkmaz problemi:* Örneğin; A kullanıcısı 1 ve 2 numaralı veri tabanından bazı bilgilere ihtiyaç duydu. B kullanıcısı da aynı veri tabanlarından aynı bilgilere erişmek istedi. A kullanıcısının sorgusu dönünceye kadar B kullanıcısının erişmek istediği 1 numaralı veri tabanı kilitlenmektedir. B kullanıcısı ise A kullanıcısının erişmek istediği 2 numaralı veri tabanının kilitlenmesine neden olur. İki kullanıcının da istediği verilere ulaşması diğerinin kayıtları serbest bırakmamasına bağlı olarak mümkün olmaz. Bu duruma çıkmaz adı verilmektedir. Farklı zamanlarda bu sorgular çalıştırıldığında bu sorun meydana gelmemektedir.

Sorunlar çözüldüğünde, dağıtık veri tabanları; performansı önemli ölçüde arttıran gelişmiş sorgu performansı, veri tabanlarından biri çökse dahi diğerinin onun yerini alabilmesi ile güvenirlik sağlanması ve yükün farklı veri tabanları ile paylaşılması ile genişletilebilirlik gibi faydaları ile tercih edilebilir olması beklenebilir.

Veri tabanı yönetim sistemlerinin yararları

Temel olarak merkezi kontrol maksatlı ihtiyaç duyulan veri tabanları bir veri tabanı yöneticisi vasıtası ile yönetilerek, veriler merkezi olarak sistemde güncellenmektedir. Veri tabanı kullanımının sağladığı avantajları şu şekilde sıralamak mümkündür [42]:

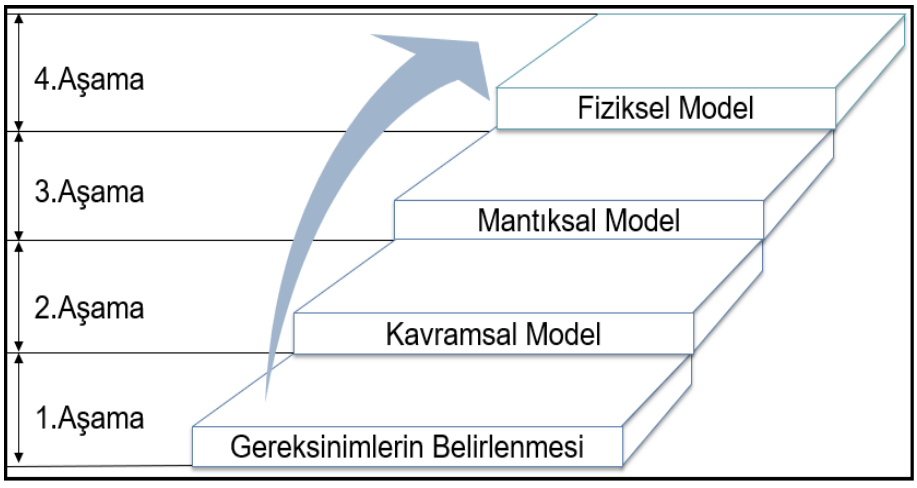
- *Veri tekrarı (data redundancy):* Aynı verilerin farklı bilgisayarlarda tutulmasını engelleyerek veri tekrarını yok eder.
- *Veri paylaşımı/eşzamanlılık (concurrency):* Bir veri tabanı yönetim sistemi (VTYS) aynı veriye saniyeler içerisinde yüzlerce defa erişilmesine imkân sağlayarak birden fazla kullanıcıya eşzamanlı hizmet sağlanabilmesine olanak tanır.
- *Veri bağımsızlığı (data independence):* Kullanıcılar veri tabanı sistemleri sayesinde kullanılan verinin yapısı veya yönetimi ile ilgilenmek zorunda kalmamaktadır.
- *Veri tutarlılığı (data consistency):* Verilerin tek bir yerde depolanarak yönetiminin sağlanması bakım maliyetini azaltır ve tutarlılık sağlanmasına katkı sağlar.
- *Veri bütünlüğü (data integrity):* Tabloların birinde yapılan bir güncelleme diğer tüm tablalarda ilave bir işleme gerek kalmaksızın güncellenir.
- *Veri güvenliği (data security):* Veri tabanı sistemleri ciddi güvenlik tedbirleri ile korunmaktadır. Yapılan uygulamalarda kullanıcılara veri tabanlarına giriş yetkisi tanınsa dahi çoğunlukla ilgili oldukları tablolara erişim izni verilerek ilave güvenlik tedbirleri oluşturulur.

Veri tabanı tasarımı

Bir veri tabanı tasarımı, farklı kullanıcıların ihtiyaçlarını karşılayabilecek şekilde konuyu oluşturmakta olan gerçeğin, veri tabanı oluşturulmasındaki gereklilik ve beklentiler kapsamında modellenip ilgili veri tabanlarına aktarımını kapsamaktadır [51].

Bir veri tabanı tasarlarken yapılması gereken ilk adım, kullanıcıların gereksinimlerini belirlemektir. Bu gereksinimler çerçevesinde veri tabanlarında bulunacak veri grupları, veri tipleri ve veri yapıları belirlenir [51]. Bu aşamadan sonra, veri tabanı yöneticisi perspektifinden modellenen gerçek, veri tabanı sistemlerinde hem bilgisayar hem de kullanıcılar tarafından anlaşılabilecek şekilde sayısal olarak tanımlanması gerekmektedir. Bu sayısal tanımlamaya "şema" adı verilmektedir. Kullanıcı düzeyleri kavramsal, bilgisayar düzeyleri ise fiziksel düzeyler olarak adlandırılmakla birlikte geleneksel tasarımda, kullanıcı düzeylerinden fiziksel düzeylere doğru tasarım gerçekleşmektedir. Kavramsal şema, bir kullanıcı açısından veri tabanı yapısını genel olarak tanımlamaktadır. Kullanıcılar bu sayede veri tabanlarının yapısını anlar ve uygulamalarını modelleyebilirler. Kavramsal şemalar, donanım ve yazılımlardan bağımsız ve kullanıcılarca anlaşılması kolay tanımlamalardır [51].

Kavramsal şemalar oldukça genel tanımlamalar olacağı için çoğunlukla doğrudan gerçekleştirilmesi mümkün olmaz. Geleneksel tasarımda, bu aşamadan sonraki adım bir veri tabanı yönetim sisteminin seçimi olmaktadır. Kavramsal veri modelinde tasarlanan kavramsal şema, veri tabanı yönetim sistemi modeli ile yeniden tanımlanmaktadır. Bu işleme mantıksal veri tabanı tasarımı adı verilmektedir [51]. Son aşamada artık fiziksel olarak sistemin kurulması sağlanır. Fiziksel veri tasarımı sonucu oluşturulan fiziksel şema ile, en yüksek verimi elde etmek için veri tabanlarının fiziki olarak nasıl organize edileceği belirlenmektedir. Şekil 3.9'da veri tabanı tasarım aşamaları gösterilmiştir.



Şekil 3.9. Veri tabanı tasarım aşamaları [51]

İlişkisel veri tabanı yönetim sistemi tasarımı

Bir ilişkisel veri tabanı tasarlanırken, genel olarak geleneksel veri tabanı tasarımında bulunan aşamalar takip edilir. Bu kapsamda, ihtiyaçlar tam olarak belirlenerek şema oluşturulur ve ilişkisel model tasarlanır. Bu adımdan sonra oluşturulan tabloların normalizasyon işlemi yapılır ve tutarlılık kuralları kontrol edilir.

İhtiyaç analizi; kullanıcı gereksinimlerinin elde edilerek veri tabanı gereksinimlerinin belirlendiği bu aşamada şu teknikler uygulanmalıdır [52]:

- Saha araştırması yapılır.
- Mevcut sorunların analizi için ortak ihtiyaçlar planlanır.
- İhtiyaç duyulan tüm belge ve formlar toplanır.
- Kullanıcı ihtiyaçlarının tam olarak anlaşılabilmesi için ilk örnekleme yapılabilir.
- Yapılan çalışma kayıt altına alınır.

ER (Entity Relationship) diyagram: Bu diyagram, yapılan veri tabanı tasarımının ifadesi için kullanılmakta olan gösterim şeklidir. Çeşitli sembollerle veri tabanı yapısı basit ve anlaşılır şekilde tanımlanabilir. İhtiyaçların analizi aşamasında elde edilen bilgiler ve bu bilgilerin kendi aralarındaki ilişkileri bir ER diyagramına dönüştürülür. Bu diyagramların nesneler ve nesne ilişkilerini basit bir şekilde açıklayabilmesi beklenir [52].

İlişkisel model: ER diyagramı ile açıklanan senaryo anlaşılır olmaktadır ancak nihai uygulama açısından bu diyagram oldukça basittir. İlişkisel model aşamasında bu diyagramlar tablolara dönüştürülür. Daha sonra ise tabloların normalizasyonu yapılmaktadır [52].

Normalizasyon: İlişkisel veri tabanı yönetim sistemlerinde tablolara karar verilmesi kuralları, veri tabanı normalizasyonu olarak tanımlanmaktadır. Bu kurallara uyma zorunlulukları uygulamaların kapsamı ve önemi ile doğru orantılıdır. Normalizasyon kuralları sayesinde verilerin tutulduğu tabloların nelerden oluşacağına karar verilmesi sağlanmaktadır [52].

Veri fazlalığı: İlişkisel veri tabanlarında aynı alanlarda birden çok veri olması ve bir tabloda aynı tip veriyi içerecek alanların varlığı veri fazlalığı sorununu oluşturmaktadır. Aynı veya farklı tabloda olabilen gereksiz veri tekrarları sistemlerin yavaşlamasına ve tutarsızlıklara sebep olabilmektedir [53].

Birinci normal form (1NF): Bir tablonun tekrarlı veriler içermesi durumuna birinci normal form (1NF) denir. Bu formda tabloların satır ve sütun kesişimlerinde sadece bir değer bulunmaktadır. Satırlar eşsiz anahtarlar ile tanımlanmaktadır [54]. Çizelge 3.2'de semt kolonlarının tekrarlaması ile 1NF özelliğini açıkça ihlal eden bir tablo örneği verilmiştir.

Çizelge 3.2. Ana tablo (normalizasyon örneği) [54]

Çalışan Ad	Soyad	Sofor	Arac	Semt 1	Semt 2	Semt 3
Orçun	Yılmaz	Ahmet	Toyota	Levent	Etiler	Ulus
Metin	Seyyar	Mehmet	Honda	Bakırköy	Ataköy	Yeşilköy
Metin	Seyyar	Tolga	Ford	Kandilli	Beylerbeyi	Kuzguncuk

Burada dikkat edilirse Semt 1, Semt 2 ve Semt 3 kolanları tekrarlayan kolonlardır. Bu tür tekrarlı verilerin bulunduğu tablolarda ekleme, silme ve güncelleme gibi sorunlar yaşanmaktadır. Bu tablo birinci normal forma uygun olabilmesi için çizelge 3.3'te görüldüğü üzere, bir kolonda birden çok veri olmayacak ve "id" sütunu ile eşsiz anahtar atanacak şekilde yeniden düzenlenmiştir.

Çizelge 3.3. Birinci normal form tablo örneği [54]

Id	Çalışan	Soyadı	Şoför	Araç	Semt
1	Orçun	Yılmaz	Ahmet	Toyota	Levent
2	Orçun	Yılmaz	Ahmet	Toyota	Etiler
3	Orçun	Yılmaz	Ahmet	Toyota	Ulus
4	Metin	Seyyar	Mehmet	Honda	Bakırköy
5	Metin	Seyyar	Mehmet	Honda	Ataköy
6	Metin	Seyyar	Mehmet	Honda	Yeşilköy
7	Metin	Seyyar	Tolga	Ford	Kandilli
8	Metin	Seyyar	Tolga	Ford	Beylerbeyi
9	Metin	Seyyar	Tolga	Ford	Kuzguncuk

İkinci normal form (2NF): Veri tabanlarının bu form yapısında olabilmesi için şu özelliklere sahip olması gerekmektedir [54]:

- Öncelikle tablonun birinci normal forma uygun olması gerekir.
- Veri tabanında anahtar olmayan değerlerin bileşik anahtar değerleri ile kısmi bağımlılık durumu oluşturulmamalıdır.
- Veri alt kümelerinin birden fazla satır içinde tekrarlamaması gerekmektedir.
- Ana tablo ile oluşturulan yeni tablolar arasında dış anahtar kullanımı ile gerekli ilişkilerin tanımlanması gerekmektedir.

Dikkat edildiğinde üçüncü ve dördüncü maddeler ikinci maddenin sonucunu oluşturmaktadır. Verilen örnekten yola çıkılacak olursa, anahtar olmayan kolonla bileşik anahtar arasında kısmi bağımlılığa bağlı veri tekrarını önlemek adına farklı tablolar oluşturulup ana tablo ile olan ilişkinin tanımlanması gerekir. Bu kapsamda; bölünen tablolar arasında ilişkiyi oluşturmak için yeni tabloda çalışan ID (CID) adlı kolon oluşturulmuştur. Bu tablonun aldığı değerin ana tablodaki eşsiz anahtara işaret etmesi durumuna dış anahtarlama denmektedir [54]. Çizelge 3.4 ve 3.5'te çalışan ve soyadı kombinasyonlarının çok tekrarlamasına bağlı ikinci normal form kapsamında ikiye bölünmüş tablolar gösterilmiştir.

Çizelge 3.4. 2NF ana tablo [54]

ID	Çalışan	Soyadı
1	Orçun	Yılmaz
2	Metin	Seyyar

Çizelge 3.5. 2NF servis tablosu [54]

CID	Şoför	Araç	Semt
1	Ahmet	Toyota	Levent
1	Ahmet	Toyota	Etiler
1	Ahmet	Toyota	Ulus
2	Mehmet	Honda	Bakırköy
2	Mehmet	Honda	Ataköy
2	Mehmet	Honda	Yeşilköy
2	Tolga	Ford	Kandilli
2	Tolga	Ford	Beylerbeyi
2	Tolga	Ford	Kuzguncuk

Üçüncü normal form (3NF): Veri tabanının bu form yapısında olabilmesi için şu özelliklere sahip olması gerekmektedir [54].

- Öncelikle tabloların ikinci normal forma (2NF) uygun olması gerekmektedir.
- Anahtar olmayan bir kolon yine anahtar olmayan başka bir kolona bağlı olmamalı ya da geçişken fonksiyonel bir bağımlılığı bulunmamalıdır. Diğer bir değişle tüm kolonlar eşsiz anahtara tam olarak bağlı olmalıdır.

Veri tabanını üçüncü normal form (3NF) kurallarına uygun hale getirmek için eşsiz anahtara bağlı olmayan kolonların yok edilmesi gerekmektedir. Çizelge 3.6, 3.7 ve 3.8'de eşsiz anahtar yerine şoför kolonuna bağlı olan araç kolonunu ayırarak dördüncü normal form özelliklerine kavuşturulan tablolar gösterilmektedir [54].

Çizelge 3.6. 3NF ana tablo [54]

ID	Çalışan	Soyadı
1	Orçun	Yılmaz
2	Metin	Seyyar

Çizelge 3.7. 3NF servis tablosu [54]

CID	SID	Semt
1	1	Levent
1	1	Etiler
1	1	Ulus
2	2	Bakırköy
2	2	Ataköy
2	2	Yeşilköy
2	3	Kandilli
2	3	Beylerbeyi
2	3	Kuzguncuk

Çizelge 3.8. 3NF şoför tablosu [54]

SID	Şoför	Araç
1	Ahmet	Toyota
2	Mehmet	Honda
3	Tolga	Ford

Üçüncü normal form (3NF) kapsamında şoför tablosu adında bir tablo oluşturularak yaratılan şoför id (SID) anahtarına servis tablosundaki kolondan referans verilmiştir [54].

Boyde-Codd normal form (BCNF): Veri tabanı yapısının bu forma uygun olduğunu anlayabilmek için bütün belirleyicilerin aday anahtar özelliği taşıması durumunun kontrol edilmesi gerekmektedir. Tablolar bir, iki ve üçüncü normal form özelliklerine göre düzenlendikten sonra bazı tabloların bütün alanları aday anahtar olarak kalabilmektedir. Bu durum söz konusu olduğunda tabloların bu forma uygun olarak parçalanması gerekmektedir [55]. Çizelge 3.9, 3.10, 3.11 ve 3.12'de tabloların bu forma uygun olarak düzenlenmesi kapsamında servis tablosunun parçalanması ve tabloların semt tablosunun değeri ile ilişkilendirilmesi görülmektedir [54].

Çizelge 3.9. BCNF ana tablo [54]

ID	Çalışan	Soyadı
1	Orçun	Yılmaz
2	Metin	Seyyar

Çizelge 3.10. BCNF servis tablosu [54]

CID	Semt
1	Levent
1	Etiler
1	Ulus
2	Bakırköy
2	Ataköy
2	Yeşilköy
2	Kandilli
2	Beylerbeyi
2	Kuzguncuk

Çizelge 3.11. BCNF semt tablosu [54]

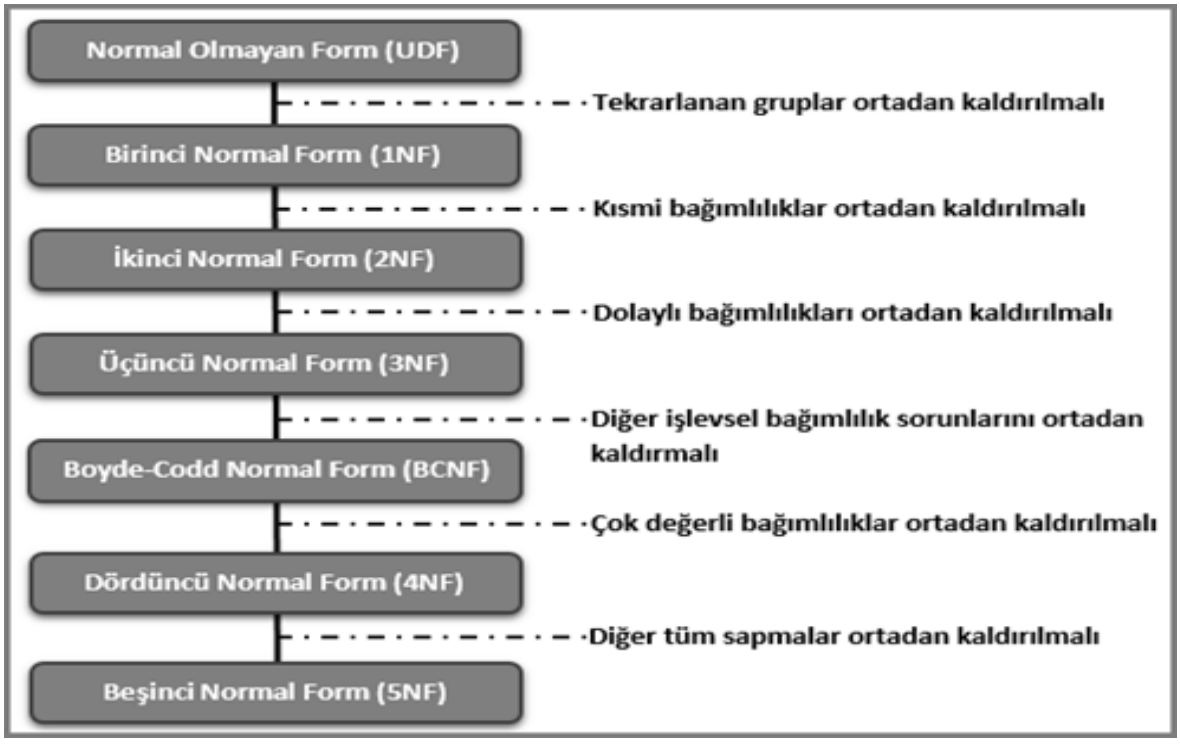
Semt	SID
Levent	1
Etiler	1
Ulus	1
Bakırköy	2
Ataköy	2
Yeşilköy	2
Kandilli	3
Beylerbeyi	3
Kuzguncuk	3

Çizelge 3.12. BCNF şoför tablosu [54]

SID	Şoför	Araç
1	Ahmet	Toyota
2	Mehmet	Honda
3	Tolga	Ford

Dördüncü ve beşinci normal form: Veri tabanı yapısının dördüncü normal forma (4NF) uygun olabilmesi için üçüncü normal form (3NF) özellikleri göstermeli ve çok değerli bağımlılıkları olmaması gerekmektedir. Yani bağımsız bire çok ilişkiler için ayrı tablolar oluşturularak birincil anahtara sahip olan alan ile birincil anahtara sahip olmayan alan arasında, birden fazla bağımsız bire çok ilişkiye izin verilmemelidir. Beşinci normal form (5NF) kapsamında her bir tablo mümkün olduğunca küçük parçalara bölünerek veri tekrarının önlenmesi gerekmektedir. Bu işlemler sonucunda, sık tekrarlama nedeni ile tutarsızlıklara neden olabilecek verilerin başka tablolara taşınması sağlanmış olacaktır [52].

Normalizasyon algoritması: İlişkisel veri tabanı yapılarının tasarımında, başlangıçta tablolar normal olmayan form özellikleri taşıyacağı için tutarlılık ve hız konularında çeşitli sorunlarla karşılaşılması durumu olabilmektedir. Bu kapsamda yukarıda bu sorunları ortadan kaldırma ve erişimi kolaylaştırma amaçlı normalizasyon kurallarına yer verilmiştir. Şekil 3.10'da bu kuralların bir başlık altında toparlandığı normalizasyon algoritması gösterilmektedir [53].



Şekil 3.10. Normalizasyon algoritması [53]

Tutarlılık kuralları: Normalizasyon kuralları gereği çoğalan tablolar ve bu tablolara bağlı olarak artan tablo ilişkileri, bu tablolar arasındaki bağlantıların düzenli yapılandırılması ihtiyacını da beraberinde getirmektedir. İlişkisel veri tabanı tasarımlarında bu ihtiyaçlara bağlı olarak uyulması gereken tutarlılık kuralları şunlardır [45]:

- Alanların atomik olması sağlanmalıdır.
- Kolonlarda bulunan "null" değeri mümkün olduğunca azaltılmalıdır.
- Birincil anahtarların basit, karalı ve kısa olması sağlanmalıdır.
- Bütün kayıtların benzersiz bir tanımlayıcısının olması sağlanmalıdır.
- Tablolarda bulunan bütün alanlar, birincil anahtarlarca tanımlanan kayıtlarla ilgili bilgi sağlamalıdır.
- Her bir tablo tek bir konuya dair veri içermelidir.
- Tablolarda yer alan bilgilerin birden fazla yerde geçmesinin önüne geçilerek veri tekrarının mümkün olduğunca azaltılması sağlanmalıdır.

Veri tabanı optimal performansı

Veri tabanlarının başlıca amaçlarından biri bilgiye çabuk erişmektir. Veri tabanı performansını iyileştirme temel sorumluluğunu taşıyan veri tabanı yöneticileri, bunu bazı ayarlamalar yaparak gerçekleştirebilmektedir. İlk adım, var olan veri tabanı hakkında bilgi toplamaktır. İkinci adım olarak optimal değişikler belirlenirken üçüncü adımda bu değişiklikler uygulanmaktadır. Son adım olarak yapılan değişiklikler kapsamında veri tabanının izlenmesi gelmektedir. Performans değerleri; veri tabanı, uygulama, sistem ve ağ seviyelerinde ayrı ayrı incelenir [45].

Veri tabanı seviyesinde performans ayarlaması: Veri tabanları, veri tabanı yöneticileri tarafından sürekli olarak incelenmelidir. Nedeni; sorunu büyümeden ve veri tabanı çökmeden belirleyebilmektir. Veri tabanlarında depolama ile ilgili bir sorun önceden belirlenmelidir. Bu işlemleri kolaylaştırmak için her veri tabanı yönetim sisteminde farklı olmak üzere izleme programları mevcuttur. Veri tabanı seviyesinde en uygun performansı sağlayabilmek için şu hususlara dikkat edilmelidir [45]:

- Veri tabanlarının ilk büyüklükleri alınarak günlük büyüme oranı belirlenir.
- Veri tabanı birden fazlaysa tamamı için bu işlem yapılır.
- Veri dosyalarının bu büyümesine paralel olarak gerekli fiziksel alan ayrılır.

Uygulama seviyesinde performans ayarlaması: Uygulama seviyesi performans ayarlaması olarak; form, rapor ve sorgu gibi veri tabanları ile etkileşim içinde olan uygulamalar düşünülmelidir. Sorgular uygulama programından yapılabileceği gibi, saklı yordamlar sayesinde veri tabanı bünyesinde de yapılabilmektedir. Bu aşamada uygulama kodlarının, veri tabanlarına etkin bir şekilde erişebilmesi için gerekli kontrollerinin yapılması sağlanmalıdır [45].

Sistem seviyesinde performans ayarlaması: Bu aşamadaki ayarlamalar veri tabanı yöneticilerinin değil, sistem yöneticilerinin sorumluluğunda yürütülmektedir. Veri tabanı yöneticilerinin üzerine düşen, sistem yöneticilerine önerilerde bulunmaktır. Bu kapsamda veri tabanı yöneticisi ve/veya sistem yöneticisinin, sistemi uygun şekilde ayarlayıp, veri tabanının aynı sistem üzerinde yer alan diğer uygulamaları olumsuz etkilemeyecek şekilde gerekli kaynaklara sahip olarak çalışmasını sağlaması gerekir [45].

Ağ seviyesinde performans ayarlaması: Veri tabanı yöneticileri uygulama alanı ile birlikte ağ performanslarını da göz önünde bulundurması gerekmektedir. Ağ ortamlarının kullanıcıların düzgün çalışmaları için yeterli olmaması, işletim sisteminde ayarlama yapılsa bile performans sorununu gidermemektedir. Veri tabanı yöneticisi uygulama ortamını değerlendirirken ağ performansına da dikkat etmelidir. Düzgün iş yükü belirlemelerinin yapılabilmesi için sağlam performans istatistikleri toplanmalı ve değerlendirilmelidir [45].

3.3. İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Yönetim Sistemleri

Günümüzde firma veya kurumların yönetim bilişim sistemlerinde çoğunlukla ilişkisel veri tabanı yönetim sistemleri kullanılmaktadır [2]. Yakın geçmişte ortaya çıkan ve hızla kullanılmaya başlanan NoSQL sistemler, birtakım dezavantajları yanı sıra ilişkisel veri tabanlarının bazı sınırlılıklarına çözüm oluşturabilmektedir.

3.3.1. İlişkisel veri tabanı yönetim sistemleri

1970'li yıllarda ortaya çıkan ilişkisel veri tabanı yönetim sistemleri, hâlihazırda en yaygın olarak kullanılan veri tabanı yönetim sistemleridir [2]. Bu sitemler satır ve sütunlardan oluşan birbirleri ile ilişkili tablolardan meydana gelmektedir. Bu sistemlerde birbiri ile eş olan kayıtların ayrılması için anahtar adı verilen bileşenler kullanılmaktadır. Bütün ilişkisel veri tabanı sistemlerinde tabloların mutlaka kayıtları diğer kayıtlardan ayıran anahtarlara sahip olması gerekmektedir. Bir ilişkisel veri tabanı yönetim sistemi tablolar üzerinde şu işlevleri sağlamalıdır [56]:

- *Seçme:* Bütün tablolarda yer alan herhangi bir bilgi veri tabanı sorguları vasıtası ile gösterilebilmelidir.
- *İz düşürme:* Bir tablonun yalnızca belli sütunlarının dahil olduğu bir seçme işlevi yapılabilmelidir.
- *Birleştirme:* Yapılan sorgu neticesinde farklı tablolarda yer alan bilgilerin ihtiyaca uygun şekilde yeri geldiğinde tek tablo gibi sunulabiliyor olmalıdır.

Bugüne kadar, ilişkisel veri tabanı sistemleri çok gelişme kaydetmiştir. Bu ilişkisel veri tabanlarının her biri kendi ilişkisel veri tabanı yönetim sistemine sahiptir. İlişkisel veri tabanları üzerinde gerçekleştirilen yönetim işlemleri bu sistemler aracılığı ile yerine getirilmektedir. Bu sistemler içerisinde; MySQL, PostgreSQL, Oracle, dBase, Informix gibi ilişkisel veri tabanı sistemleri başta sayılabilmektedir. Bu sistemler aracılığı ile sağlanan işlemleri şu şekilde sıralamak mümkündür [57]:

- Veri tabanına görüntü veya fonksiyon eklenmesi.
- Veri tabanına alan, indeks veya yeni tablo eklenmesi.
- Veri tabanına ait performansın takip edilmesi ve yönetilmesi.
- Veri tabanın yedeklenmesinin sağlanması.
- Veri tabanının yeterli güvenliğinin sağlanması ve yetkisiz kişilerce erişimine engel olunması.

MySQL veri tabanı yönetim sistemi özellikleri

Bu tezde, bütün ilişkisel veri tabanlarının tek tek açıklanması mümkün değildir. İlişkisel ve NoSQL veri tabanları performanslarının karşılaştırıldığı bu çalışmada ilişkisel veri tabanı yönetim sistemi olarak MySQL seçildiği için bu sistemin özelliklerine yer verilmiştir.

MySQL, açık kaynak kodlu web yazılımıdır. MySQL; altı milyondan fazla sisteme yüklenmiş, çok kullanıcılı, çoklu iş parçacıklı, sağlam ve hızlı bir veri tabanı yönetim sistemi olarak tanımlanmaktadır. Bu veri tabanı yönetim sistemi ilk defa 1994 yılında geliştirilmeye başlanmış ve ilk sürümü 1995'te yayınlanmıştır. 2001, 2004 ve 2008 yıllarında yeni sürümü yayımlanan sistem 2009 yılında Oracle firması tarafından satın alınmıştır. C++ ve C dillerinde yazılan ve 1995'ten itibaren aktif olarak kullanılmakta olan MySQL sistemi, halihazırda en yaygın kullanılan ilişkisel veri tabanı yönetim sistemlerinden biridir. Bu denli yaygın olmasındaki en büyük etkenlerin başında yazılımının kolay kullanıma sahip olması ve Windows, OS/2 ve UNIX platformlarında ücretsiz olarak dağıtılması gelmektedir. Açık kaynak kodlu olan bu veri tabanı yönetim sisteminin farklı platformlar için ayrı ayrı hazırlanmış indirilebilir sürümleri mevcuttur [58]. MySQL veri tabanı sisteminin kabiliyetleri aşağıdaki gibi sıralanabilmektedir [59]:

- *Güçlü karakter seti desteği:* Bütün yazılım dillerine uygun olarak tasarlanmış olan bu sistem, Türkçe de dahil olmak üzere alfabeler arası geçiş kolaylığı da sağlamaktadır.
- *Kolay sorgulama altyapısı:* Geniş komut dizinlerini barındıran ve bu dizinler ile ihtiyaca uygun çok çeşitli sorgular yapılabilen geniş bir kütüphaneye sahiptir. Komut dizinlerinin ayrıntılı açıklamalarına internet üzerinden kolaylıkla ulaşılabilmektedir.
- *Sorguların önbellekte tutulması:* Öncesinde yapılmış olan belirli sayıda sorgu önbellekte kaydedilir ve sorgu tekrar kullanıldığında işlemciye az yük biner.
- *Kod sistemleri ile uyumlu çalışabilmesi:* Java, ASP, Perl ve PHP gibi tercih edilen gelişmiş yazılımlar, bu veri tabanı yönetim sistemi ile uyumlu olarak çalışabilmektedir.

3.3.2. İlişkisel olmayan (NoSQL) veri tabanı yönetim sistemleri

İlişkisel olmayan veri tabanları ilk defa 1998'de Carlo Strozzi tarafından tasarlanmıştır. SQL arayüz kullanılmayan bu sisteme tasarımcısı, "No Relation" ilişkisel olmayan anlamı ifade eden bir isim koyulması gerektiğini ileri sürmüştür [60]. NoSQL kavramının isimlendirilmesinde bazı sorunlar yaşanmıştır. "NonRel" veya "NoRDBMS" gibi ifadeler tercih edilmemiştir. NoSQL tabiri için akla gelen ilk anlam "SQL değil" olmaktadır. Ancak bu ifade için "not only SQL" tabiri de kullanılmaktadır. Nedeni ise, SQL sorgu yapılarına aşina uygulama geliştiricileri için sorgulama mantığının adapte edilmesidir [61].

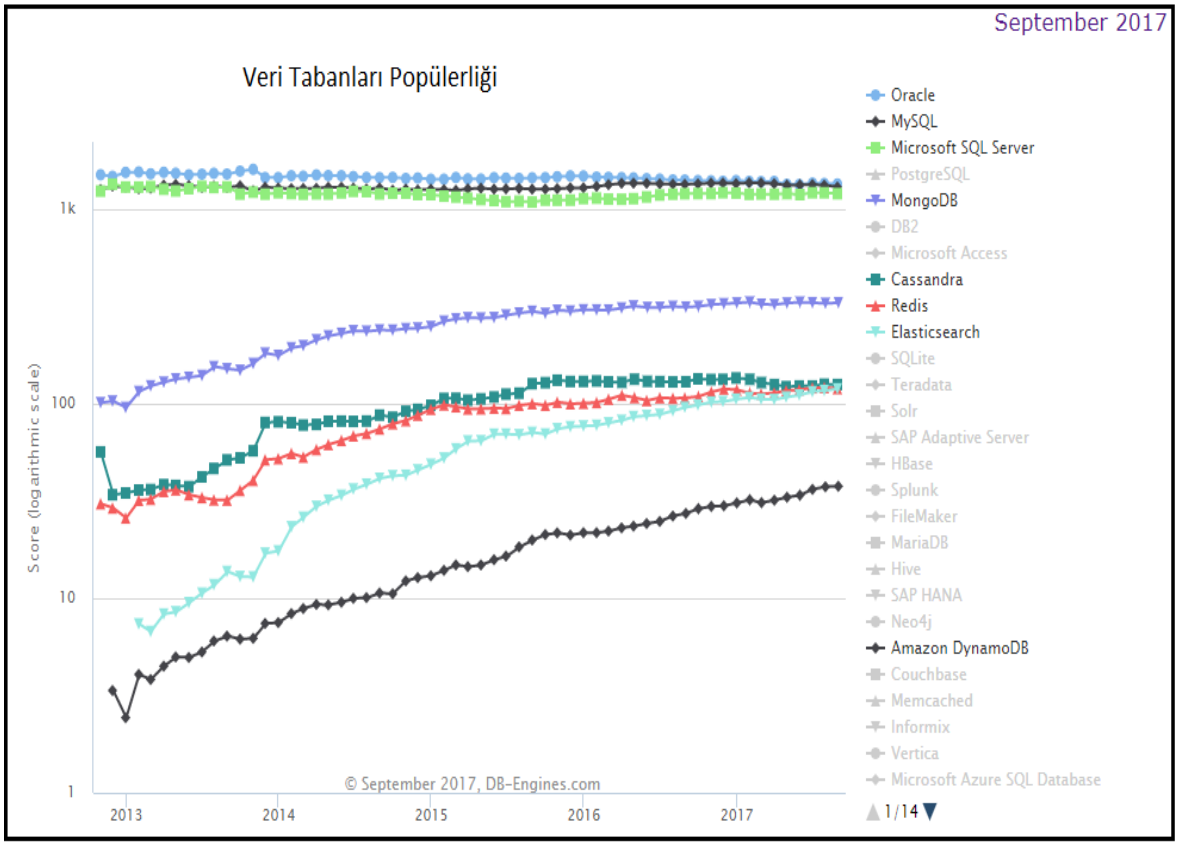
İlişkisel olmayan veri tabanı sistemleri, Eric Evans tarafından 2009 yılında planlanan dağıtık veri tabanı toplantısı ile tekrar kullanılmaya başlanmış ve Atlanta'daki konferans NoSQL üzerinde derin etkilere sebep olmuştur [62]. Yakın geçmişte ortaya çıkan bu sistemler sağladıkları avantajların fark edilmesi ile hızla kullanılmaya başlanmıştır. MongoDB, HBase, CouchDB, Cassandra, Redis, Voldemort, Hypertable, Dynomite gibi veri tabanı sistemleri, NoSQL veri tabanlarından bazılarını oluşturmaktadır.

<u>NoSQL sistemlerin veri tabanı pazarındaki yeri</u>

İnternetin yayılması, bilişim sistemlerinin hayatın her alanına nüfus etmesi ve buna paralel olarak artan veri miktarı veri tabanı sistemlerinde de gelişime ihtiyaç duyulmasına neden olmuştur. İlişkisel veri tabanı yönetim sistemlerine (RDBMS) alternatif olarak ortaya atılan bu sistemler birtakım dezavantajları yanında geleneksel veri tabanı modellerinde karşılaşılan bazı problemlere çözüm olmuştur [61].

Bu sistemlerin ortaya çıkmasının en önemli sebebi, internet ortamında yapısal olmayan veri miktarındaki hızlı artışa paralel olarak ortaya çıkan performans sorununa çözüm oluşturabilmektir. Halihazırda ilişkisel veri tabanı yönetim sistemlerini kullanan firmalar, artan veri ve veri tabanı kısıtlaması problemlerini, ekonomik imkanlar dahilinde daha fazla önbellek kullanarak aşmaya çalışmaktadır. Bu çözümlerin yeterli olmadığı durumlarda NoSQL veri tabanı sistemleri devreye girmiştir [61].

Şekil 3.11'de ilişkisel ve NoSQL veri tabanları yönetim sistemleri içerisinden en çok kullanılan modellerin ele alınarak, yıl yıl kullanım miktarlarının gösterildiği veri tabanı yönetim sistemleri kullanımı grafiği verilmiştir [2].



Şekil 3.11. Veri tabanları kullanım grafikleri [2]

Şekil 3.11'de görüldüğü üzere yapılan sorgulamada, birçok alternatifleri arasından yaygın olarak kullanılması hususu dikkate alınarak, ilişkisel veri tabanı yönetim sistemleri olarak; Oracle, MySQL ve Microsoft SQL Server sistemleri, NoSQL veri tabanı yönetim sistemleri olarak ise; MongoDB, Cassandra, Redis, Elasticsearch ve Amazon DynamoDB sistemleri seçilmiştir.

Şekil 3.11 ile ilgili yapılan analizde; yaygın kullanılan ilişkisel veri tabanı yönetim sistemlerinin, artan kullanıcı sayısına rağmen yıllar içerisinde artmamakla birlikte pek fazla bir değişim göstermediği gözlemlenmiştir. Bunun yanında yaygın olarak kullanılan NoSQL veri tabanı yönetim sistemlerinde ciddi artışlar meydana gelmiştir. Bu kapsamda NoSQL veri tabanlarının veri tabanı pazarına hızla girdikleri söylenebilmektedir.

NoSQL veri tabanı sistemlerinin sınıflandırılması

Çok farklı özellik ve teknoloji kullanan NoSQL veri tabanı sistemleri literatürde çeşitli şekillerde sınıflandırılabilmektedir [63]. Literatürde yapılan detaylı araştırma neticesinde bu sistemlerin genel olarak anahtar değer tabanlı, doküman tabanlı, sütun tabanlı ve grafik tabanlı olmak üzere dört yapı tipi şeklinde toparlanabildiği gözlemlenmiştir. Şekil 3.12'de, yaygın olarak kullanılmakta olan NoSQL veri tabanı sistemleri, yapı tipleri ile beraber gösterilmiştir [64].



Şekil 3.12. Bazı NoSQL veri tabanlarının sınıflandırılması [64]

1. Anahtar değer tabanlı NoSQL veri tabanları: Anahtar değer tabanlı veri tabanı yapıları, verileri anahtar ve değer çifti ile depolar. Bu, oldukça basit ve temel NoSQL veri tabanı türüdür ancak etkin ve güçlü bir model sağlarlar. Bu modelde, veriler anahtar ve değer çiftlerinde saklanır. Bu yapı, uygulama geliştiricilerine daha az şema ve esneklik sağlayan bir model tasarlama olanağı tanır. Veri tabanlarındaki anahtarlar dizine eklenmiştir. İlişkisel veri tabanı yönetim sistemlerinden farklı olarak bu veri tabanları, her satırın formatının farklı olabileceği en esnek veri modeli sunar. Anahtar değer tabanlı NoSQL veri tabanları, yüksek yeniden ölçeklenebilirlik sağlar; bu durum onları yeniden tasarlamaya gerek kalmadan boyutlarının arttırılabileceği anlamına gelir. Bu sistemler son derece hızlıdır ve büyük yükleri işlemek üzere tasarlanmıştır [64].

Bu sistemler daha çok, kullanıcı oturum bilgilerini yönetmek ve saklamak ile kullanıcının alışveriş sepeti bilgilerini saklamak için kullanılmaktadırlar [64]. Örnek anahtar değer tabanlı veri tabanları arasında Cassandra, Amazon DyanmoDB, Azure Table Storage (ATS), Riak ve BerkeleyDB bulunmaktadır.

Güçlü yönleri: Esnek şema modeli, son derece hızlı işlemci, muazzam veri yüklerini yönetme kapasitesidir.

Zayıf yönleri: Düşük tutarlılık, tabloların birleşmesi ve toplanmasının yapılamamasıdır.

2. Doküman/belge tabanlı NoSQL veri tabanları: Belge tabanlı veri tabanları, verileri belge biçiminde depolar. Her belgeye, belgeyi almak için kullanılan benzersiz bir anahtar atanır. Anahtarın veri türü string, numeric veya array şeklinde de depolanabilmektedir. Belgeler, JSON, XML ve bunun gibi standart biçimlerde saklanır. Belge tabanlı bir veri tabanındaki belgelerin kaydı, ilişkisel veri tabanlarındaki kayıtlara benzer. Ancak belge tabanlı NoSQL veri tabanlarının şema yapısı çok daha esnektir [64]. Belge / doküman tabanlı veri tabanlarına örnek olarak MongoDB, CouchDB ve Couchbase gösterilebilmektedir.

İlişkisel veri tabanında, bir tablodaki her kayıt aynı sütunlara sahiptir ve bir kayıt için veri bulunmayan sütunlar boş (null) olacaktır. Buna karşılık, belge tabanlı veri tabanında, her kaydın farklı şemaları olabilir ve bu boş sütunları saklamaya gerek olmamaktadır [64].

Bu veri tabanları öncelikle belge odaklı bilgilerin saklanması, alınması ve yönetilmesi için tasarlanmıştır. Bu tip veri tabanı yapıları daha çok şu amaçları yerine getirmek için kullanılmaktadır [64]:

- Blog uygulaması ve yorum işlevselliği sağlanması.
- Yarı yapılandırılmış veya yapılandırılmamış büyük miktarda kütük depolanması.
- Makine / sensör tarafından üretilen verilerin saklanması.
- Anket uygulaması için veri yönetimi ve verilerin saklanması.
- Farklı ürün portföylerini içeren uygulama desteği sağlanması.

Güçlü yönleri: Esnek Şema, hızlı anahtar değeri erişimi, büyük yazma performansı ve ölçeklenebilirliktir.

Zayıf yönleri: Daha az tutarlılık sağlamaları ve karmaşık sorgulama eksikliğidir.

3. Grafik tabanlı NoSQL veri tabanları: Grafik teorisine dayanan bu veri tabanları, farklı düğümler arasındaki ilişkileri veya bağlantıları keşfetmek için tasarlanmıştır. Bu veri tabanları, verileri temsil etmek ve depolamak için kenarları ve düğümleri kullanır. Düğümler nesne olarak gösterilir ve kenarlar, bu düğümler arasındaki ilişkiler gibi davranır. Her düğüm ve kenar, anahtar değer çiftleri biçiminde ek özellikler depolayabilir. Grafik veri modeli, muazzam miktarda veriyi hızlı ve verimli bir şekilde döndürebilme konusunda etkilidir. Grafik veri tabanı şemaları, yarı yapılandırılmış verinin daha az ve verimli bir şekilde depolanmasını sağlar [64]. Grafik veri tabanı, nesnelerin birbirine bağlı olduğu uygulamalar için yararlı olabilmektedir. Bu uygulamaların en iyi örneği Facebook ve diğer Sosyal medya uygulamalarıdır. Grafik tabanlı sistemlere örnek olarak Neo4J, Titan veri tabanları verilebilir. Grafik tabanlı veri tabanları daha çok şu uygulamalarda kullanılmaktadır [64]:

- Sosyal medya ve ağ uygulamaları,
- Ağ ve bulut yönetimi,
- Güvenlik ve erişim kontrolü,
- Tavsiye motorları.

4. Sütun tabanlı NoSQL veri tabanları: Bu veri tabanları; verileri, veri tablolarının satırları olarak saklamak yerine, veri sütunlarının bölümleri olarak depolama için tasarlanmıştır. Tüm sütunlar ayrı ayrı izlenir ve tek sütunun değerleri birlikte saklanır. Geniş sütunlu depolarda verileri depolamak çok yüksek performans ve yüksek derecede ölçeklenebilir bir mimari sunar. Saklanan sütun, az sayıda sütun gerektiren sorguların performansını büyük ölçüde artırabilir. Sütun tabanlı veri tabanları, verileri tablolarda saklamak yerine çoklu dağınık mimaride depolarlar. Bunlar, birçok makineye dağıtılan verileri depolamak ve işlemek için yaratılmıştır [64].

Sütun tabanlı yapılar temelde Google büyük tablo uygulamasına dayanır [64]. Bu modellere örnek olarak HBase, BigTable ve HyperTable gösterilebilmektedir.

Güçlü yönleri: Büyük veriler için yüksek verim sağlaması, ölçeklenebilirlik ve güçlü bölümleme desteği sağlamasıdır.

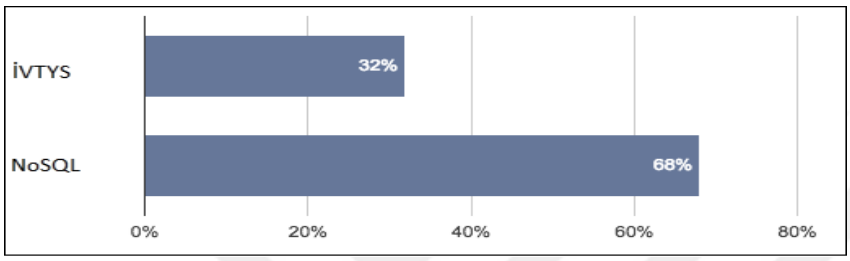
Zayıf yönleri: Karmaşık sorgular yürütememesidir.

NoSQL sistemleri kullanım maksatları

İlişkisel veri tabanlarına oranla daha yeni bir teknoloji olan NoSQL sistemler, henüz firmaların yönetim bilişim sistemlerince geliştirilen uygulamalarda daha az bir kullanıma sahiptir. Kullanım oranı göreceli olarak az olan ancak her gecen gün artan bu sistemlerin birçok özellikleri yanı sıra birtakım sınırlılıkları mevcuttur [65]. Vaish [66]'da NoSQL veri tabanı yönetim sistemlerinin temel avantajları ve kullanım maksatlarını dört temel başlık altında ele almıştır:

- *Şablonsuz veri gösterimi:* İlişkisel veri tabanı yönetim sistemlerinde, önce tablo ve sütunlar oluşturulur ve sonrasında veriler satırlara sıra ile kaydedilir. Eğer tanımı olmayan bir alana veri kaydı yapmak istenirse yeni bir sütun eklemek gerekmektedir. Bu durum bütün tabloyu etkileyebilecek olumsuzlukları da beraberinde getirebilmektedir. NoSQL veri tabanı yönetim sistemlerinde ise veri tabanı hiyerarşinde esneklik söz konusudur.
- *Geliştirilme zamanı:* Bu sistemlerde karmaşık sorgular kullanılması tercih edilmemektedir. Uygulama geliştiricilerinin karmaşık sorgulara ihtiyaç duymayacağı uygulamalarda bu sistemleri tercih etmesi geliştirme zamanını bakımından avantaj sağlar.
- *Hızlılık:* NoSQL sistemlerinin en büyük avantajlarından biri hızlı olmasıdır.
- *Ölçeklenebilirlik (scalability):* Bu sistemler yatay olarak ölçeklenebilmektedir. Sürekli artan RAM ihtiyacı doğuran uygulamalarda bu veri tabanı sistemlerinin kullanılması ciddi maliyet avantajını beraberinde getirmektedir.

Günümüzde; veri merkezi izleme, finans veri analizi, sensor gibi zaman serisi veri uygulamaları hızla artış kaydetmektedir [67]. Yapılan son araştırmalar bu zaman serisi uygulamalarında veri tabanı olarak NoSQL sistemlerin ilişkisel veri tabanı sistemlerine tercih edildiğini göstermiştir [67]. Şekil 3.13'te "zaman serisi uygulamalarında hangi veri tabanı motorunu kullanıyorsunuz?" araştırmasına verilen cevaplar kapsamında, kullanılan veri tabanı yönetim sistemlerinin ilişkisel ve NoSQL dağılımlarının yüzdeleri gösterilmiştir [68].



Şekil 3.13. Zaman serisi uygulamalarında veri tabanı teknolojilerinin kullanım oranları [68]

Şekil 3.13 ile gösterilen çalışmada ilişkisel veri tabanı yönetim sistemleri olarak**;** MySQL, MariaDB Server ve PostgreSQL sistemleri seçilmiştir. NoSQL veri tabanı sistemleri olarak ise; Elastic, InfluxDB, MongoDB, Cassandra, Couchbase, Graphite, Prometheus, ClickHouse, OpenTSDB, DalmatinerDB, KairosDB, RiakTS sistemleri ele alınmıştır.

MongoDB veri tabını sistemi özellikleri

NoSQL veri tabanı olarak çok sayıda seçenek vardır. Bu tezde bütün NoSQL veri tabanı sistemlerini tek tek açıklamak mümkün değildir. İlişkisel ve NoSQL veri tabanları performanslarının karşılaştırıldığı bu çalışmada MongoDB veri tabanı sistemi seçildiği için bu sistemin özelliklerine yer verilmiştir.

MongoDB sistemi, doküman taban tipinde, c++ yazılım dili ile geliştirilmiş bir NoSQL veri tabanı sistemidir. Hız ve ölçeklenebilirlik özellikleri ile ön plana çıkan bu sistemin kullanım alanları; analiz verisi, web içerik, yorum saklama ve yönetme ile yüksek hacimli problemlerdir [69].

MongoDB veri tabanı sistemi 2007 yılında geliştirilmeye başlanmıştır. 2009 yılında lisansı alınmıştır. İlk defa 2011 yılı mart ayında kullanıma sunulmuştur. The New York Times, Uber, Foursquare gibi çok sayıda büyük şirket tarafından kullanılmaktadır [70]. MongoDB veri tabanı sistemlerinin; dinamik veri yapısı gerektiren, arşiv ve veri izi (log) tutan ve yüksek hacimli veri okuması ve yazması gereken uygulamalarda kullanılması daha uygun olmaktadır [71].

Bu sistemde yapısal veriler JSON benzeri dokümanlarda (BSON = Binary JSON) tutulmaktadır. BSON belgeleri; içerisindeki kayıtların sıralı haldeki listesinden oluşan nesnelerdir. Her bir kayıt, bir alan adıyla belirlenen tipteki değerden oluşmaktadır. MongoDB sistemlerinde; klasik ilişkisel veri tabanlarından farklı olarak, tablo yerine koleksiyon, sütun yerine alan ve satır yerine BSON doküman kavramları kullanılmaktadır. İlişkisel veri tabanlarında bir tablo içerisindeki sütunlar aynı olmak zorundayken, bu veri tabanı sisteminin koleksiyonu içerisindeki farklı dokümanlar farklı alanlarda barındırılabilmektedir [63]. MongoDB veri tabanı sisteminin diğer NoSQL sistemler ile karşılaştırıldığında, öne çıkan özelliklerini şu şekilde özetlemek mümkündür [71].

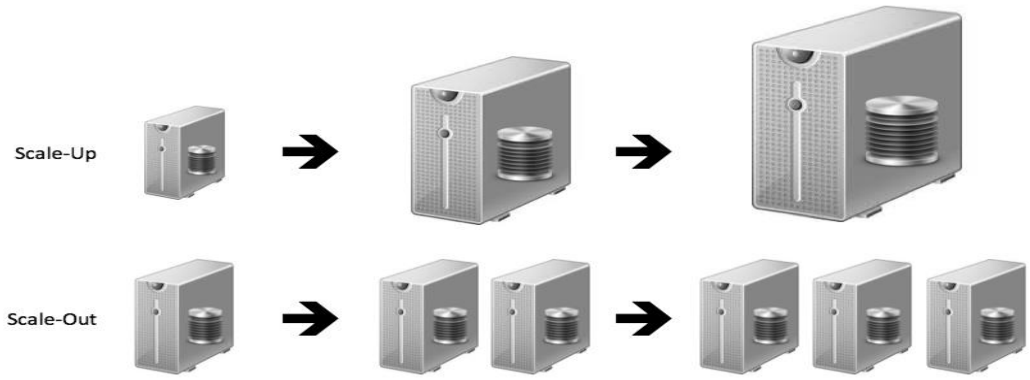
- *Sorgu desteği:* Birçok NoSQL veri tabanı sistemi yalnız anahtar değer üzerinden sorgulama yapmaya olanak tanırken bu sistemler, ilişkisel veri tabanı sistemlerine benzer, istenilen alana düzenli ifadeler ile sorgu yazabilme imkânı sağlamaktadır.
- *Performans:* İstenen alana sorgu yapabilen bu sistem aynı zamanda bu alanlara indeks oluşturmak sureti ile daha yüksek performans sağlayabilmektedir.
- *Ölçeklenebilirlik:* MongoDB; “master-slave replication” olarak adlandırılan sistem ile ölçeklenebilirlik sağlamaktadır. Bu ölçeklendirme sisteminde yazma işlemi “master” sunucudan, okuma işlemi ise “slave” sunudan yapılmaktadır. “Master” sunucuya bir nedenle erişilemezse “slave” sunuculardan biri yeni “master” sunucu olarak devreye girmekte ve sistem bu sayede kesintisiz olarak çalışabilmektedir.
- *Sharding:* Bu özellik büyük hacimdeki verinin sunucular arasında paylaştırılmasını sağlamaktadır. Bu sayede yükün dağıtılması sağlanmaktadır.
- *Sürücü desteği:* Basit bir kütüphane tasarımına sahip bu sistem; C, C++, C#, Java, PHP, Python, Erlang gibi birçok yazılım dili için sürücü desteği sağlamaktadır.

3.3.3. İlişkisel ve NoSQL veri tabanı yönetim sistemlerinin karşılaştırılması

İlişkisel ve NoSQL veri tabanları, dikey ve yatay ölçekleme, veri yapısı çeşitleri, ACID ve BASE kavramları ile okunan ve yazılan şema parametreli kullanılarak karşılaştırılmıştır.

Dikey (Scale-up, vertical sacale) ve yatay ölçekleme (scale out, horizontal scale)

Büyük veri, sosyal ağlar ve bilişim uygulamalarındaki kullanıcı sayılarının her geçen gün hızla artması veri tabanlarındaki okuma ve yazma isteklerini ciddi şekilde arttırarak ilişkisel veri tabanlarının bazı durumlarda tek başına yetersiz kalmalarına neden olabilmektedir. Birkaç yıl öncesine kadar performansı arttırmanın iki seçeneği bulunmaktaydı. Birincisi, donanım kaynaklarının arttırılması (scale-up), diğeri ise performans eniyileme (performance tunning) uygulamaları desteğinin sağlanması idi. Günümüzde performans gelişimi için üçüncü bir seçenek olan yatay ölçekleme (scale-out, horizontal scacle) kavramları ortaya çıkmıştır. Şekil 3.14'te de dikey ölçekleme (scale-up) ve yatay ölçekleme (scale-out) kavramlarını açıklayan gösterim sunulmuştur.



Şekil 3.14. Yatay ve dikey ölçekleme (scale-up ve scale-out) kavramları [72]

Dikey ölçekleme (scale-up, vertical sacale): İlişkisel veri tabanları dikey ölçekleme (scale-up) özeliklere sahip veri tabanlarıdır. Scale-up kavramını bir örnek üzerinden açıklamak gerekirse, veri tabanı bir kutu gibi düşünüldüğünde veri büyüdükçe veri tabanının da büyümesi gerekecektir. Çok büyük veriler için çok büyük kutulara ihtiyaç duyulmuş olacaktır. Bu şekilde geleneksel büyüme ile ram ve işlemci gücü arttırılarak sağlanacak avantaj ve dezavantajlar şunlardır [72]:

Avantajları:

- Tek sunucu olacağı için birden fazla sunucuya göre harcanan enerji daha düşük olacaktır.
- Tek cihazı soğutma maliyeti, yatay ölçekleme için kullanılacak birden fazla cihaza oranla nispeten az olacaktır.
- Tek lisans anahtarı gerekli olacaktır.
- Bazı durumlarda ağ trafiği daha az olabilmektedir.

Dezavantajları:

- Mevcut bir donanımın geliştirilmesi (dikey ölçekleme) benzer özelliklerde farklı donanımlar eklenmesine (yatay ölçeklendirme) oranla ciddi maliyeti beraberinde getirecektir.
- Donanımsal bir sorun çıkması durumunda, tek cihaz olması nedeni ile tüm sistemi etkileyecektir.

Yatay ölçekleme (scale-out, horizontal scale): İlişkisel olmayan veri tabanları yatay ölçekleme özelliklerine sahip sistemleridir. Yatay ölçekleme kavramı verilen örnek üzerinden açıklanmak istenirse; bu şekilde büyüyen veri tabanlarında, yeni kutu eklenerek büyüme sağlanmaktadır. Teorik olarak eklenebilecek kutuların sınırı olmadığı için büyüyebilecek verinin de bir sınırı olmayacaktır. Daha az işlemcili ve ram kapasiteli makinaların eklenmesi ile gerçekleşen büyümenin, avantaj ve dezavantajları şu şekilde gerçekleşmektedir [72]:

Avantajları:

- Dikey büyümeye oranla maliyet çok daha az olmaktadır.
- Bir node'un çökmesi ayrık yapı sayesinde sistemin tamamının devre dışı kalmasını engeleler.

Dezavantajları:

- Lisans maliyeti artacaktır.
- Sunucu makinaları ile doğru orantılı olarak elektrik ve soğutma maliyeti artacaktır.
- Daha fazla ağ cihazlarına ihtiyaç duyulmaktadır.

Veri yapıları (structured, semi structured and unstructured data)

İlişkisel veri tabanlarında veriler, “structured data” olarak tanımlanan veri yapıları şeklinde tutulmaktayken NoSQL veri tabanlarında veriler ya yarı yapısal ya da tamamen yapısal olmayan şekilde saklanmaktadır.

İlişkisel veri tabanı yapıları (structured data): İlişkisel veri tabanlarında veriler, veri yapıları (structured data) şeklinde tutulmaktadır. Bu sistemler içerisinde çok sayıda mantıksal nesne bulunmaktadır. Bunlar arasında belli başlı olarak; saklanmış yordam, görüntü, tetikleyici ve kısıtlamalar sayılabilmektedir. Bu nesnelerin temel amacı, veri tabanına giriş yapan verilerin tutarlılıklarını sağlamaktır. Bu nesneler sayesinde veri tabanına hatalı bir kayıt girişi tespit edilerek gerekli uyarıların verilmesi sağlanmaktadır. İndeks; tablolarda tutulan verilerin hızlı bir şekilde erişilebilmesi ve sorgulanabilmesi için kullanılan nesnelerdir. Bir veri tabanında çok sayıda tablo olabileceği gibi, bir tablo içerisinde farklı niteliklerde indeksler de olabilmektedir. Her veri tabanında kendi içerisinde oluşturulan nesneleri tutmaya yarayan, sistem tablo ve şemaları yer almaktadır. Bu yapılara “Meta-Data” adı verilmektedir [57].

NoSQL veri yapıları (semi-structured and unstructure data): NoSQL veri tabanlarında veriler yarı yapısal ve tamamen yapısal olmayan şekilde saklanmaktadır. Örneğin bir metin, web sayfası, öz geçmişi hatta video veya resimler gibi veriler, herhangi bir sınıflandırmaya veya özellik çıkarımına gerek olmaksızın kullanılabilir. Bu verilerin meta-datası olup olmadığı da önemli olmamaktadır. İnternet ortamında yarı yapısal ve yapısal olmayan veri miktarının yapısal veriye oranla çok daha hızlı bir şekilde artması NoSQL veri tabanı yönetim sistemlerinin önemini arttırmaktadır.

ACID ve BASE kavramları

İlişkisel veri tabanları ile NoSQL veri tabanları; uyulması gerekli olan farklı kurallara göre hareket etmektedir. İlişkisel veri tabanları ACID kısaltması ile ifade edilen kurallar bütününü kullanırken, NoSQL veri tabanları BASE kısaltması ile ifade edilen kuralları kullanmaktadır.

ACID (Atomicy, Consistency, Isolation, Durability) kavramı: İlişkisel veri tabanı yönetim sistemleri varlığı kesin olan verileri saklamak, ihtiyaç olduğunda onlara ulaşmak ve onları yönetmek üzere tasarlanmışlardır. Bu sistemler için önemli olan verinin her zaman doğru ve tutarlı bir şekilde saklanabilmesi ve ulaşılabilir olmasıdır.

İlişkisel veri tabanı yönetim sistemlerinde "iş süreci" (transaction) adı verilen işlemler gerçekleştirilmektedir. İş süreçlerinde, bir işlem bitmeden diğer işlemin başlamasına izin verilmez. İşlemin yarıda kesilmesine de izin verilmez. Mesela bir banka transferi yapılırken o transfer bitmeden banka hesaplarına erişim engellenir. Veri tabanı mimarilerinde bu şekilde bir "iş süreci" gerektiren durumlarda uyulması gerekli olan prensiplere ACID denir. İlişkisel veri tabanı yönetim sistemleri üzerinde; select, insert, update ve delete gibi bir dizi operasyonun gerçekleşebilmesi için uygulanması gereken kurallar bütünü olan bu prensipler; "bölünmezlik, tutarlılık, izolasyon ve dayanıklılık" kelimelerinin İngilizce karşılıklarının ilk harflerinden oluşan "ACID" kısaltması ile ifade edilmektedir [73]. ACID kuralları [74]:

- *Atomicity (bölünmezlik):* Bu kural iş süreci (trancsaction) işleminin ana özelliğidir. Operasyondaki bütün işlemler başarılı olmak zorundadır. Yarım kalan iş süreci (trancsaction) bloğu veri tutarsızlığına neden olmaktadır. Bu nedenle ya bütün işlemler gerçekleştirilir. Ya da iş sürecinin gerçekleştirdiği tüm değişiklikler geri alınarak o işlem yok sayılmış olur.
- *Consistency (tutarlılık):* İşlem sonrası elde edilen çıktıların tutarlığını ifade eder. Bir iş sürecinde (trancsaction) güncelleme yapıldığı taktirde geriye kalan bütün işlemlerde de bu güncelleme yapılmalı ya da ilgili güncelleme geri alınmalıdır.
- *Isolation (izolasyon):* Operasyon esnasında dış müdahalelere izin verilmez. İşlemler birbirinden izole olarak çalıştırılır. Bir iş sürecinin (trancsaction) ancak bütün işlemleri gerçekleştikten sonra, diğer iş süreçleri tarafından görülebilmesine izin verilir.
- *Durability (dayanıklılık):* Eğer operasyon esnasında bir hata meydana gelirse sistem, işlemlerin bütününü güvence altına alabilmek için dayanıklı olmak zorundadır.

BASE (Basically Available Soft-state services with Eventual-consistency) kavramı: NoSQL veri tabanı yönetim sistemleri tutarlılıktan ziyade hız faktörü ile öne çıkmaktadır. Bu modelde bazı yöntemler ile tutarlılık sağlansa bile asıl ön plana çıkan husus taleplere en kısa sürede cevap verebilecek hız kavramıdır. Sosyal medya üzerinden bir örnek vermek gerekirse, bir cihazda bulunan uygulama üzerinden arkadaş olarak eklenen kişi, başka bir cihaz üzerinden kontrol edildiğinde henüz eklenmemiş olarak görülme ihtimali olasıdır. Ancak bir süre sonra bakıldığında bu tutarlılığın sağlandığı görülecektir. Yani isteklere hızlı bir şekilde cevap veren bu sistem anlık veri tutarlılığını garanti etmemektedir. Ancak bir süre sonra tüm veriler tutarlı hale gelecektir. Tutarlılık sağlanana kadar geçen sürede sunucuya bir talep iletilirse sunucu elinde o anda hangi bilgi varsa o bilgiyi istemciye iletecektir [75].

İlişkisel veri tabanlarının kullandığı "ACID" kurallarına karşın NoSQL veri tabanı sistemleri; "kolay ulaşılabilirlik, esnek durum ve eninde sonunda tutarlılık" kelimelerinin İngilizce karşılığı olan; "basic availability, soft state ve eventualy consistency" kelimelerinin ilk harflerinden oluşan "BASE" kısaltması ile ifade edilen kuralları kullanmaktadır [76]. BASE kuralları [75]:

- *Kolay ulaşılabilirlik (basic availability):* İlişkisel olmayan veri tabanı sistemleri hemen her talebe cevap verebilmektedir. Fakat bu cevap verme zorunluluğu bir hata durumunda dahi geçerliliğini sürdürdüğü için verinin tutarlılığı garanti edilemez. Kısacası verilerin o anki tutarlılığından feragat edilerek kolay ulaşılabilirlik sağlanmış olur.
- *Esnek durum (soft state):* NoSQL sistemlerde veriler, diskte kalıcı olmayan şekilde tutulmaktadır. Anlık tutarlılığın olmaması, verilerin bütün cihazlarda aynı olması zorunluluğunu ortadan kaldırmaktadır. Esnek durum kapsamında zaman içinde değişikliklerin olabileceği varsayılır.
- *Eninde sonunda tutarlı (eventualy consistency):* İşlem etkileri sistem durumuna göre tanımlanamayan bir zaman sonra diğer cihazlara yansır. İlişkisel olmayan sistemlerin gelecek bir zaman diliminde tutarlı olacağı var sayılır. Anlık tutarlılık garanti edilmese de eninde sonunda tutarlılık sağlanmaktadır.

Yazılan (scheme on write) ve okunan şema (schema on read)

1970'li yıllarda ilişkisel veri tabanı sistemlerinin ortaya çıkmasından buyana verilerin analiz edilmek üzere depolanmasında, yazılan şema prosedürü kullanılmaktadır. Ancak son zamanlarda büyük veri ve NoSQL veri tabanlarının revaçta olması ile okunan şema prosedürlerine olan ilgi de artmıştır.

Yazılan şema (scheme on write): İlişkisel veri tabanı yönetim sistemlerince kullanılan yazılan şema, toplanan verinin veri tabanına yazılmadan önce bir şema oluşturulması olarak tanımlanmaktadır. Yazılan şema yapılarında verilerin toplanmasından sonra ilk görev tabloların oluşturulması ve ilişkilerin tanımlanması ve nihayet şemanın hazırlanmasıdır. Bu prosedürde veri tabanına herhangi bir veri girişi yapılmadan önce verilerin nasıl kaydedileceğinin belirleneceği ve şema adı verilen yapıların oluşturulması şarttır [77].

Okunan şema (schema on read): Yapılandırılmamış verilerin veri tabanına yazılmasına müsaade eden prosedürler, okunan şema olarak adlandırılmaktadır. NoSQL veri tabanı sistemleri bu prosedürleri kullanmaktadır. Verileri depolamadan önce şemanın tanımlanması gerekmediği için anında yeni veri kaydını mümkün kılarak çeviklik sağlamaktadır. Yapılandırılmamış verilerin artış hızının yapılandırılmışlara oranla çok daha fazla olması bu prosedürleri önemli kılmaktadır [77].

4. UYGULAMA: İLİŞKİSEL VE İLİŞKİSEL OLMAYAN (NOSQL) VERİ TABANI YÖNETİM SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

Bilgi sistemleri kapsamında yaşanan değişim ve gelişim, verilerin modellenmesi ve belli bir modelle ifade edilen verilerin saklanmasını sağlayan veri tabanı sistemlerinin kullanımını zorunlu kılmaktadır.

Yüzlerce kullanıcıya hizmet sağlayan temel bir kurum veya firmanın yönetim bilgi sistemlerince geliştirilen uygulamalardan, milyonlarca kullanıcısı olan çok büyük web uygulamalarını geliştiren organizasyonlara kadar farklı alanlarda farklı önceliklere göre veri modelleme ve depolama gerekliliği ortaya çıkmıştır. Verinin büyüklüğü, miktarı ve karmaşıklığı gibi etkenlere bağlı olarak farklı veri modelleme, veri depolama ve sorgulama yöntemleri geliştirilmiştir.

Geliştirilen birçok ilişkisel veri tabanı yanı sıra yakın geçmiş itibarıyla ortaya çıkan çok sayıdaki NoSQL veri tabanları, farklı özelliklerine bağlı avantaj ve dezavantajları nedeni ile, kurum veya firmaların yönetim bilgi sistemlerinde çalışan veri tabanı yöneticilerinin hazırlanan/hazırlanacak uygulamalara uygun bir veri tabanı yönetim sistemini seçme problemini doğurmuştur.

Bu çalışmada, kurum veya işletmelerin yönetim bilgi sistemlerinde hazırlanan uygulamalarda kullanılan veri tabanı yönetim sistemlerinin seçimi problemi üzerinde durulmuştur.

Firmaların yönetim bilgi sistemlerince çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanı performansının karşılaştırılması, yeni geliştirilecek uygulamalarda kullanıcılara veri tabanı yönetim sistemleri secimi konusunda fikir verilmesi, NoSQL veri tabanı yönetim sistemlerinin sağladığı bilinen hız avantajının firmaların mevcut uygulamalarda değişime gitmesini gerektirecek boyutlarda olup olmadığının araştırılması hedeflenmiştir.

Yol haritası

İlişkisel ve NoSQL veri tabanı yönetim sistemlerinin performans karşılaştırması kapsamında sırasıyla şu çalışmalar yapılmıştır;

- Kıyaslanacak veri tabanı sistemlerinin seçilmesi,
- Kullanılan veri tabanı sunucu sistemlerinin belirlenmesi,
- Veri tabanı şemaları oluşturulması,
- Basitten zora doğru veri tabanı sorgularının belirlenmesi,
- Ölçümler ve ölçüm metrikleri bilgilerinin belirlenmesi,
- Testlerin yapılması, sonuçların kaydedilmesi ve elde edilen sonuçların yorumlanması çalışmaları gerçekleştirilmiştir.

4.1. Seçilen Veri Tabanı Yönetim Sistemleri ve Seçim Nedenleri

Firma veya kurumların yönetim bilgi sistemlerinde çalışan veri tabanı yöneticilerine, geliştirilen uygulamalarda kullanılabilecek farklı veri tabanı sistemleri arasından doğru olanı seçebilmelerine yardımcı olacak performans testi uygulamasında, ilişkisel veri tabanı yönetim sistemi (RDBMS) olarak MySQL, NoSQL veri tabanı yönetim sistemi olarak ise MongoDB sistemi kullanılmıştır.

Çalışmaya başlamadan önce detaylı araştırma yapılarak temel bir kurum veya firma yönetim bilgi sistemlerinde kullanılan genel uygulamalar için kullanılabilecek en uygun ilişkisel ve NoSQL veri tabanı yönetim sisteminin seçilmesi hedeflenmiştir.

İlişkisel veri tabanı yönetim sistemi olarak MySQL sistemi seçilmesinde; kurum veya firmalarca en çok kullanan üç sistemden biri olarak [2] yaygın olarak kullanılıyor olması, diğer ilişkisel modellerde olduğu gibi ACID işlevselliği sağlaması, kolay sorgulama altyapısı ve web üzerinde tecrübeye dayanarak yazılmış çok sayıda onaylı çözüm vasıtası ile kullanıcıların karşılaştıkları sorunları kolayca çözebilmelerine olanak sağlayan dokümantasyona sahip olması etkili olmuştur.

İlişkisel olmayan (NoSQL) veri tabanı yönetim sistemi olarak MongoDB seçilmesinde; kurum veya firmalarca en çok kullanılan NoSQL veri tabanı sistemi olması [2], basit bir kütüphane tasarımına sahip olması, C, C++, C#, Java, PHP, Python, Erlang gibi birçok yazılım dili için sürücü desteği sağlaması, JOIN işlemleri dışında zengin sorgu desteği sağlaması, doküman tabanlı olması ve en önemlisi diğer NoSQL veri tabanlarından farklı olarak çok daha detaylı ve kullanım kolaylığı sağlayan dokümantasyon desteği sağlaması etkili olmuştur.

4.2. Veri Tabanı Sunucu Sistemleri Özellikleri

Bu uygulamada bir adet fiziksel sunucu üzerinde kurulu iki adet sanal veri tabanı sunucu sistemi kullanılmıştır. Sunucu sistemleri yapılandırmaları aynı özelliklere sahiptir. Çizelge 4.1'de sunucu sistemlerine ait teknik özellikler yer almaktadır.

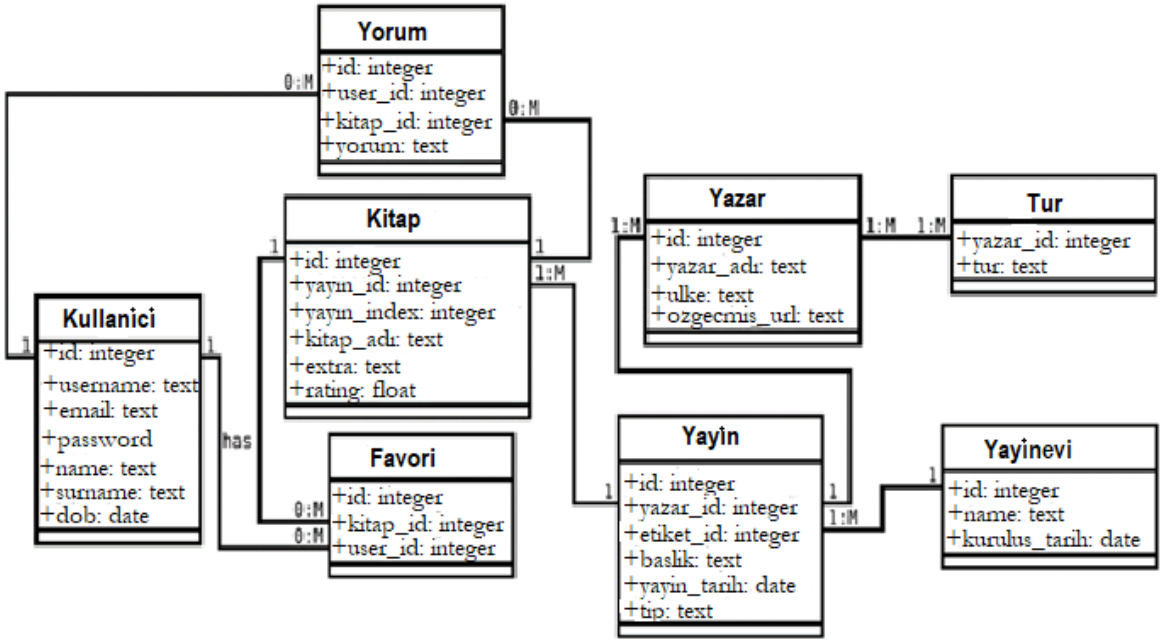
Çizelge 4.1. Fiziksel ve sanal sunucu sistemleri teknik özellikleri

Fiziksel Sunucu Özellikleri (Cisco C460 M2)	İşlemci	4 x Intel Xeon E7-4870
	RAM	64 GB
	Chipset	Intel 7500 Chipset
	Disk	2 x 146 GB RAID 1 (İşletim Sistemi) 4 x 300 GB RAID 5 (Veri Depolama Alanı)
Sanal Sunucu Özellikleri (Hyper-V Temel Ayarlar)	İşlemci	1 x Intel Xeon E7-4870
	RAM	16 GB RAM
	Disk	100 GB (İşletim Sistemi) 150 GB (Veri Depolama Alanı)

4.3. Veri Tabanı Şeması

İlişkisel ve NoSQL veri tabanlarının test edileceği bu projede, firma veya kurumların yönetim bilgi sistemlerince sıklıkla geliştirilen uygulamalar baz alınarak, okuyucuların zevk ve tercihlerine göre kitap tavsiyesi yapabilen bir kütüphane uygulaması geliştirilmiştir. Bu uygulamadan, yapılan çalışmada sadece veri tabanı sorgulamalarının yapılabilmesi için faydalanılmıştır. Söz konusu uygulama çerçevesinde biri MySQL, diğeri ise MongoDB veri tabanı olmak üzere iki adet veri tabanı tasarlanmıştır. Mümkün olduğunca eşit şartlar sağlanabilmesi için ilişkisel veri tabanı tasarımında normalizasyon adımları uygulanmıştır. Tasarlanan her iki veri tabanı şemalar ayrı ayrı gösterilmiştir.

Şekil 4.1'de yapılacak çalışma kapsamında hazırlanan ilişkisel (MySQL) veri tabanı şeması gösterilmiştir.

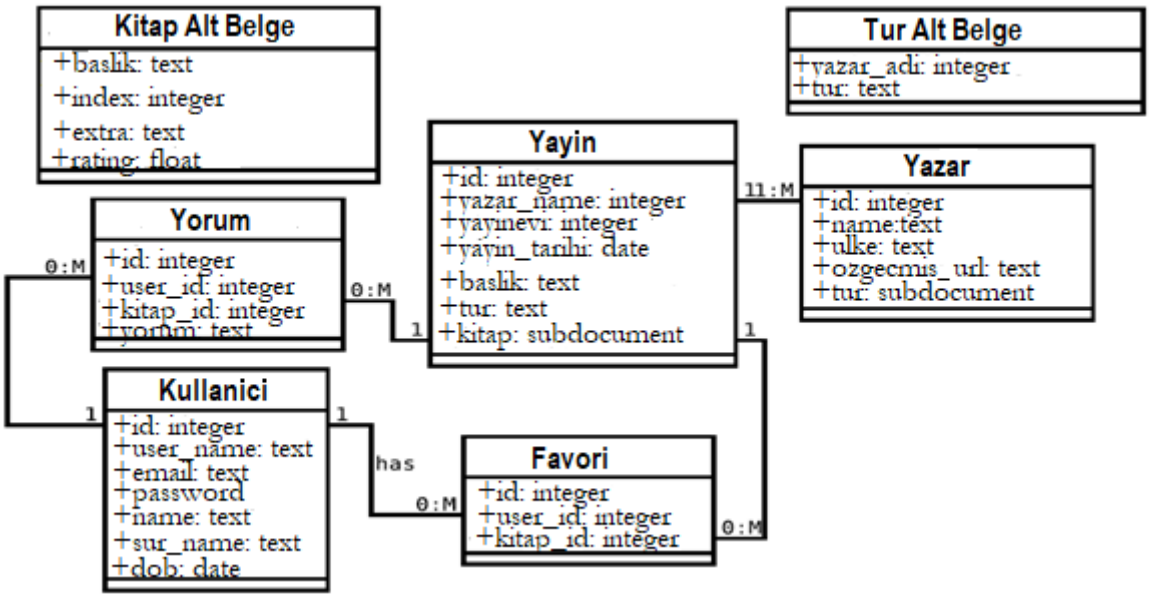


Şekil 4.1. İlişkisel (MySQL) veri tabanı şeması

Diğer NoSQL veri tabanlarında olduğu gibi MongoDB veri tabanı sistemi için de herhangi bir şema oluşturulması zorunluluğu yoktur. Konuyu açmak gerekirse; MongoDB gibi NoSQL sistemlerde ilişkisel veri tabanı yönetim sistemlerden farklı olarak, koleksiyon yapıları (ilişkisel veri tabanı yönetim sistemlerindeki karşılığı tablo) çok daha esnektir. Koleksiyona yeni bir alan (ilişkisel veri tabanı yönetim sistemlerindeki karşılığı sütun) eklemesi veya çıkarılması bir engel teşkil etmez. Bu sistemlerde her doküman ayrı bir varlıktır ve kendine özgü yapısı olabilmektedir. Aynı zamanda bir doküman bir başka alt doküman da içerebilir. Bu alt tablolar ilişkisel veri tabanlarındaki tablolar arası ilişkileri temsil etmektedir.

Bu çalışmada ilişkisel veri tabanı için hazırlanan şema, denormalizasyon olarak da tabir edilen tabloların bir araya getirilmesi ile alt belgelerden yararlanarak MongoDB veri tabanı şeması olarak revize edilmiştir. Şema incelendiğinde yayın koleksiyonunda; yayınevine ait bilgilerin yer alması yanı sıra kitaplar tablosunun yerini tutan kitap alt bilgisi indeksinin yer aldığı kolaylıkla görülebilmektedir. Aynı şekilde yazar koleksiyonunda tür tablosunun yerini tutan tür alt bilgisi indeksi yer almaktadır.

Şekil 4.2'de aynı uygulama için hazırlanan NoSQL (MongoDB) veri tabanı şeması gösterilmektedir.



Şekil 4.2. NoSQL (MongoDB) veri tabanı şeması

4.4. Veri Tabanı Sorguları

Bu çalışmada, geliştirilen uygulamalarda karşılaşılabilecek farklı veri tabanı sorgularını örneklendirebilmek için, her iki veri tabanı içinde aynı sonucu döndürecek dört farklı veri tabanı sorgusu oluşturulmuştur. Tasarlanan ilişkisel (MySQL) veri tabanı için; birinci sorgu olarak sadece "SELECT" deyimi içeren basit bir sorgu hazırlanmıştır. Tablo birleştirmenin test edildiği ikinci sorgu için "INNER JOIN" içeren birinci sorguya nispeten karmaşık bir sorgu hazırlanmıştır. Üçüncü sorgu için ise INNER JOIN değimi içermeyen fakat iç içe "SELECT" içeren karmaşık sorgu hazırlanmıştır. Son olarak dördüncü sorgu için hem "INNER JOIN" hem de iç içe "SELECT" değimi içeren daha karmaşık bir sorgu hazırlanmıştır.

Farklı sorgu çeşitleri kullanılmasının nedeni; firma veya kurumların yönetim bilgi sistemlerince hazırlanarak, yöneticilerin karar vermelerine destek olan uygulamaların raporlanmasında, veri tabanlarına uygulanacak sorgu çeşitliliğine veri tabanlarının göstereceği reaksiyonu ölçümlemektir.

MongoDB veri tabanı sisteminde daha önce de belirtildiği üzere ilişkisel veri tabanlarındaki JOIN benzeri bir karşılık yoktur. Bu farklı mimari yapı nedeni ile, yapılacak sorgular sadece bir koleksiyon üzerinde çalışmaktadır. Aşağıda MySQL veri tabanı yönetim sistemi için kullanılan SQL sorgu türleri ile birlikte MongoDB veri tabanı sistemi için kullanılan sözdizimi karşılıkları şekiller ile gösterilmiştir.

Sorgu 1: Sadece SELECT değimi içeren basit sorgu

```
SELECT *
FROM Kullanici
WHERE username = 'username'
```

Şekil 4.3. Sadece SELECT değimi içeren basit sorgu için SQL sorgu dili

```
db.kullanici.find({
    "username": "username"
});
```

Şekil 4.4. Sadece SELECT değimi içeren basit sorgu için MongoDB sözdizimi

Sorgu 2: Sadece bir INNER JOIN içeren karmaşık sorgu

```
SELECT 'Favori.kitap_id' AS fKID , 'Favori.user_id' AS fUID
FROM Favori AS b INNER JOIN Favori AS a
ON b.user_id = a.user_id
WHERE a.kitap_id = 123456 AND a.user_id != 987654
```

Şekil 4.5. Sadece bir INNER JOIN içeren karmaşık sorgu için SQL sorgu dili

```
db.Favori.find({
    "Favori.kitap_id": 123456,
    "Favori.user_id": {
        "$ne": 987654
    }
});
```

Şekil 4.6. Sadece bir INNER JOIN içeren karmaşık sorgu için MongoDB sözdizimi

Sorgu 3: İç içe SELECT içeren karmaşık sorgu

```
SELECT 'Favori.kitap_id' AS fKID , 'Favori.user_id' AS fUID
FROM Favori
WHERE user_id IN
    (SELECT 'Favori.user_id' AS fid
    FROM Favori
    WHERE kitap_id = 123456 AND user_id != 987654)
```

Şekil 4.7. İç içe SELECT içeren karmaşık sorgu için SQL sorgu dili

```
db.Favori.find({
    "user_id": {
        "$in": ["Favori.kitap_id = 123456",
                "Favori.user_id != 987654"]
    }
});
```

Şekil 4.8. İç içe SELECT içeren karmaşık sorgu için MongoDB sözdizimi

Sorgu 4: İç içe SELECT ve INNER JOIN içeren karmaşık sorgu

```
SELECT 'Kitap.yayin_id' AS kID , 'Yayin.id' AS yID
FROM Kitap INNER JOIN Yayin ON Kitap.yayin_id = Yayin.id
WHERE yazar_id  IN
    (SELECT 'Tur.yazar_id' AS tYID
    FROM Tur AS c
    INNER JOIN Yazar AS d
    ON c.yazar_id = d.id  WHERE d.yazar_adi = 'yazar_adi' )
```

Şekil 4.9. İç içe SELECT ve INNER JOIN içeren karmaşık sorgu için SQL sorgu dili

```
db.Yayin.find({
    "$where":"this.Kitap.yayin_id == this.Yayin.id",
    "yazar_id": {
        "$in": ["Tur.yazar_id = Yazar.id"]
    },
    "Yazar.name": "yazar_adi"
});
```

Şekil 4.10. İç içe SELECT ve INNER JOIN içeren karmaşık sorgu için MongoDB sözdizimi

4.5. Ölçümler ve Ölçüm Metrikleri

Veri tabanlarının karşılaştırmasında, yapılan işlemlerin zamanı tek tek ölçülmüş ve kayıt altına alınmıştır. Bu işlem için Clock() fonksiyonundan yararlanılmış ve belirli bir süre CPU üzerinde harcanan zaman ile sonuçlarının elde edilmesi sağlanmıştır.

Yapılan uygulamada öncelikle, kurum veya firmaların yönetim bilgi sistemleri kısmında çalışan veri tabanı yöneticilerinin, geliştirilen uygulamalar için tercih ettiği ilişkisel ve NoSQL veri tabanı yönetim sistemlerinin farklı sorgu türlerine göre nasıl performans gösterdikleri teste tabi tutulmuştur. Daha sonra ilişkisel ve NoSQL veri tabanı sistemlerinin veri ekleme ve veri silme performansları test edilmiştir. Yapılan testlerde sorgu kayıt sayısı ve zaman parametreleri kullanılmıştır. Her iki veri tabanından elde edilen sonuçların, şekiller ile gösterimi sağlanmıştır. Son olarak ortaya çıkan sonuçlar veri tabanı performansı kapsamında analiz edilmiştir.

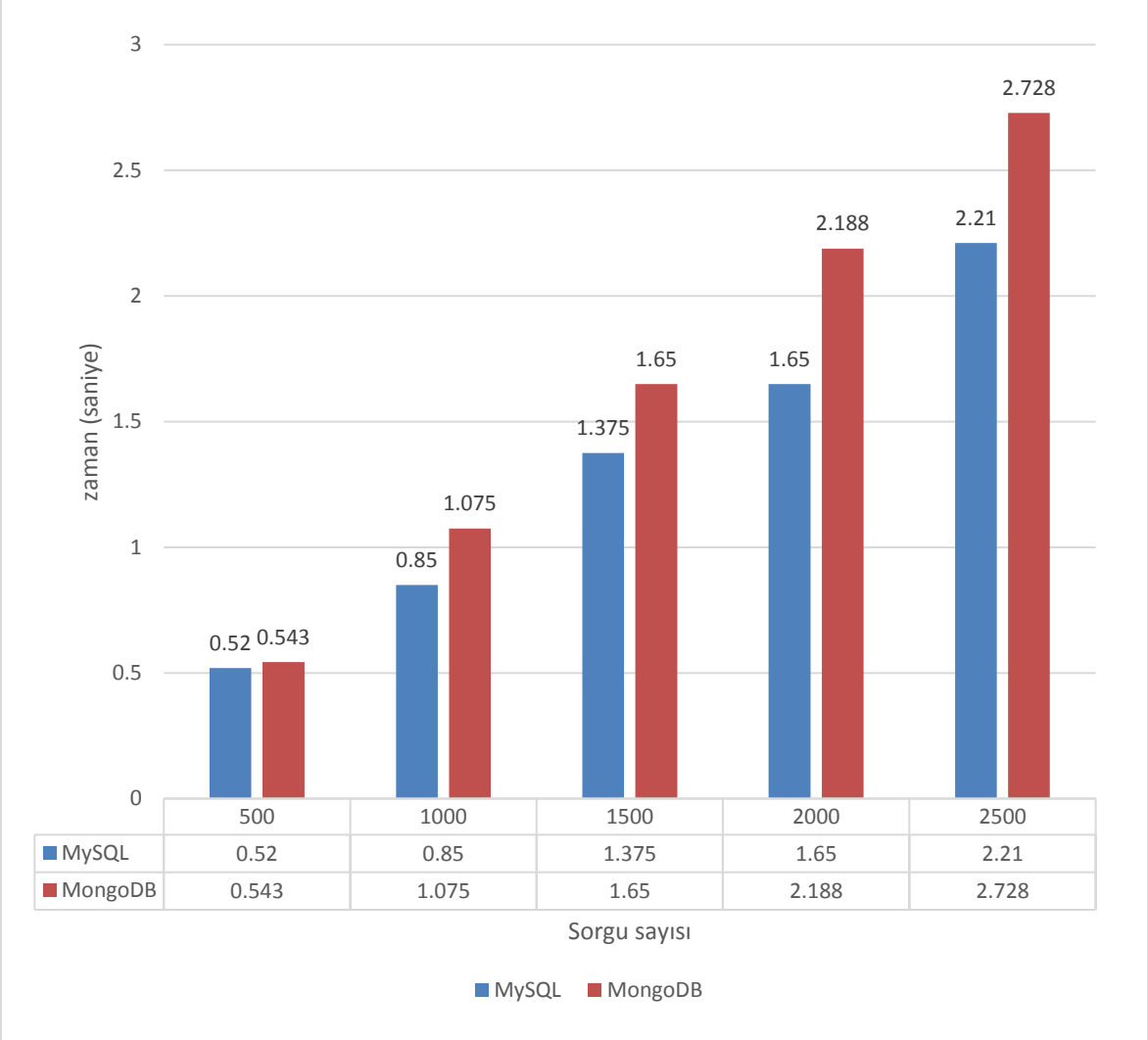
Yapılan çalışmada daha önce açıklanan koşullar kapsamında veri tabanlarının detaylı olarak karşılaştırılabilmesi için çok çeşitli durumlar yaratılmak istenmiştir. Ölçüm için kullanılan yapılandırmalarda sorgu kayıt sayısı olarak 500 ile başlanmış, 500'ün katları olacak şekilde testlere devam edilmiş ve genel olarak doğrusal bir grafik elde edildiği için 2500 satırda kesilmiştir. 500 kayıt sayısı ile başlanmasının nedeni ise yönetim bilgi sistemlerinde geliştirilen 500'den daha az kayıt sayısına sahip uygulamaların veri tabanı yapısından kaynaklı bir performans problemi yaşaması ihtimalinin bulunmamasıdır.

Veri tabanı sorgularının performans testleri bittikten sonra veri ekleme ve silme performansları da test edilmiştir. Veri ekleme ve silme işlemleri genellikle tek satır üzerinden yapılmasına rağmen performans farkını tam olarak ortaya koyabilmek ve kıyaslamanın sağlıklı olabilmesi için teste 100 kayıt sayısı ile başlanmıştır. İlgili testler 500, 1 000, 5 000, 10 000 ve 50 000 kayıt için tekrarlanmıştır. Elde edilen test sonuçları doğrusal bir eğilim göstermesi nedeni ile test edilen kayıt sayısı 50 000 kayıtta bitirilmiştir.

Elde edilen sonuçların mümkün olduğunca sağlıklı olabilmesi için; veri tabanı sistemlerinin aynı sorguyu RAM üzerinden çalıştırmasını engellemek üzere her seferinde bilgisayar kapatılıp açılmış ve CPU değerleri dikkate alınarak açıldıktan sonra eşit süre beklenmiştir. Her bir ölçüm beşinci testten sonra ortalama değerlerden fazla bir sapma olmaması nedeni ile beş adet test ile bitirilmiştir. Her test sonucunda sorgulardan her birini gerçekleştirmek için alınan ortalama süreler hesaplanarak raporlanmıştır. Veri tabanı sorguları bölümünde yer alan sorgu koşullarına göre çalıştırılan bu sorgular, MySQL ve MongoDB veri tabanları sistemlerinin her ikisine eşdeğer miktarda gerçekleştirilmiştir.

4.6. Sonuçlar ve Analiz

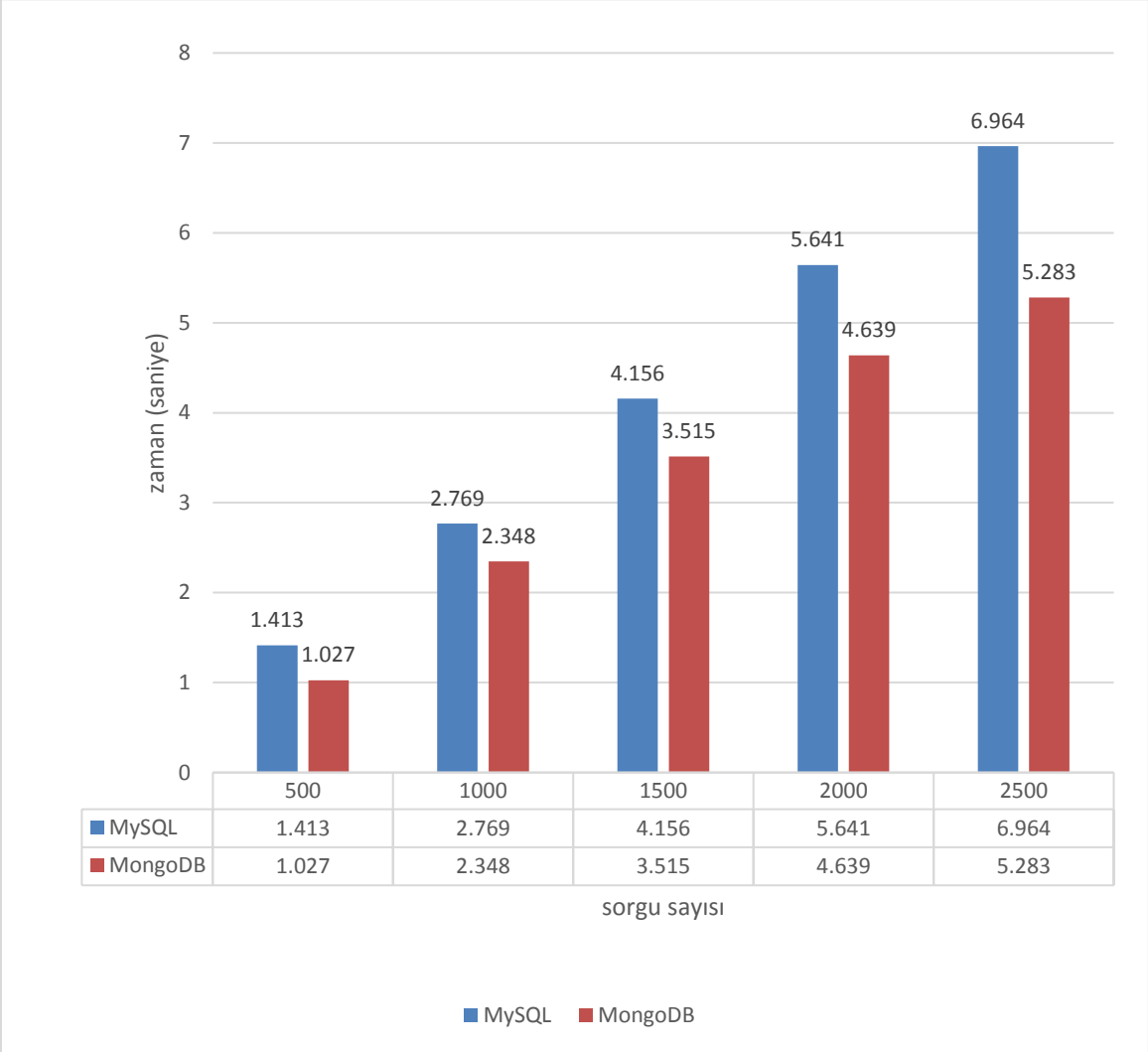
MySQL ve MongoDB veri tabanlarına öncelikle birinci sorgu olan basit sorgu ile karşılaştırma testi uygulanmıştır. Elde edilen sonuçlar şekil 4.11'de gösterilmiştir.



Şekil 4.11. Sorgu 1 kullanılarak veri tabanı sistemlerinin performans karşılaştırması

Şekil 4.11 ile ilgili yapılan analizle; sadece SELECT komutu yer alan bu sorguda, ilişkisel veri tabanı olan MySQL veri tabanı sisteminin MongoDB sistemine göre %20 civarı performans avantajı sağladığı kolaylıkla gözlemlenebilmektedir. Bu durum NoSQL veri tabanı yönetim sistemlerinin bilinen hız avantajının, tablo birleştirmeyi gerektirmeyen veri tabanı sorguları içeren uygulamalarda geçerli olmadığını, aksine bu şekildeki sorgulamalarda ilişkisel veri tabanı sistemlerinin daha hızlı olabildiğini ortaya koymuştur.

Şekil 4.12'de; MySQL ve MongoDB veri tabanlarına ikinci sorgu olarak tanımlanan "INNER JOIN" terimi içererek tablo birleştirmeyi gerektiren karmaşık sorgu kodu ile uygulanan karşılaştırma testi sonuçları gösterilmiştir.

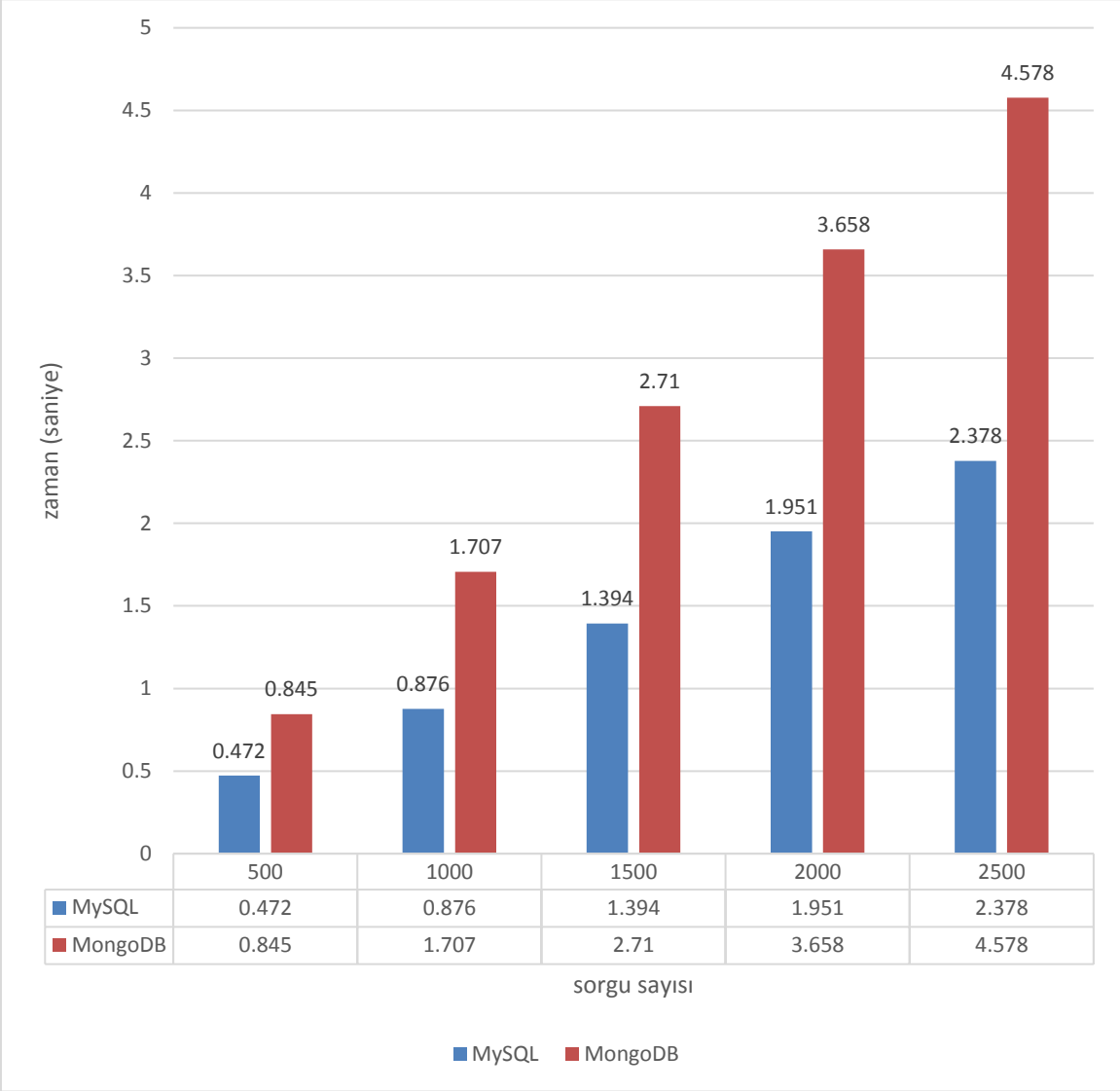


Şekil 4.12 Sorgu 2 kullanılarak veri tabanı sistemlerinin performans karşılaştırması

Şekil 4.12 ile ilgili yapılan analizde; MySQL veri tabanı sisteminin MongoDB sistemine göre daha kötü bir performans gösterdiği gözlemlenmiştir. MongoDB veri tabanı sisteminin bu sorgu türündeki performans avantajı %20 civarında gerçekleşmiştir.

Bu durum tablo birleştirmeyi gerektirecek çok karmaşık olmayan veri tabanı sorguları içeren uygulamalarda NoSQL veri tabanlarının, ilişkisel veri tabanlarına göre performans avantajı sağladığını göstermiştir.

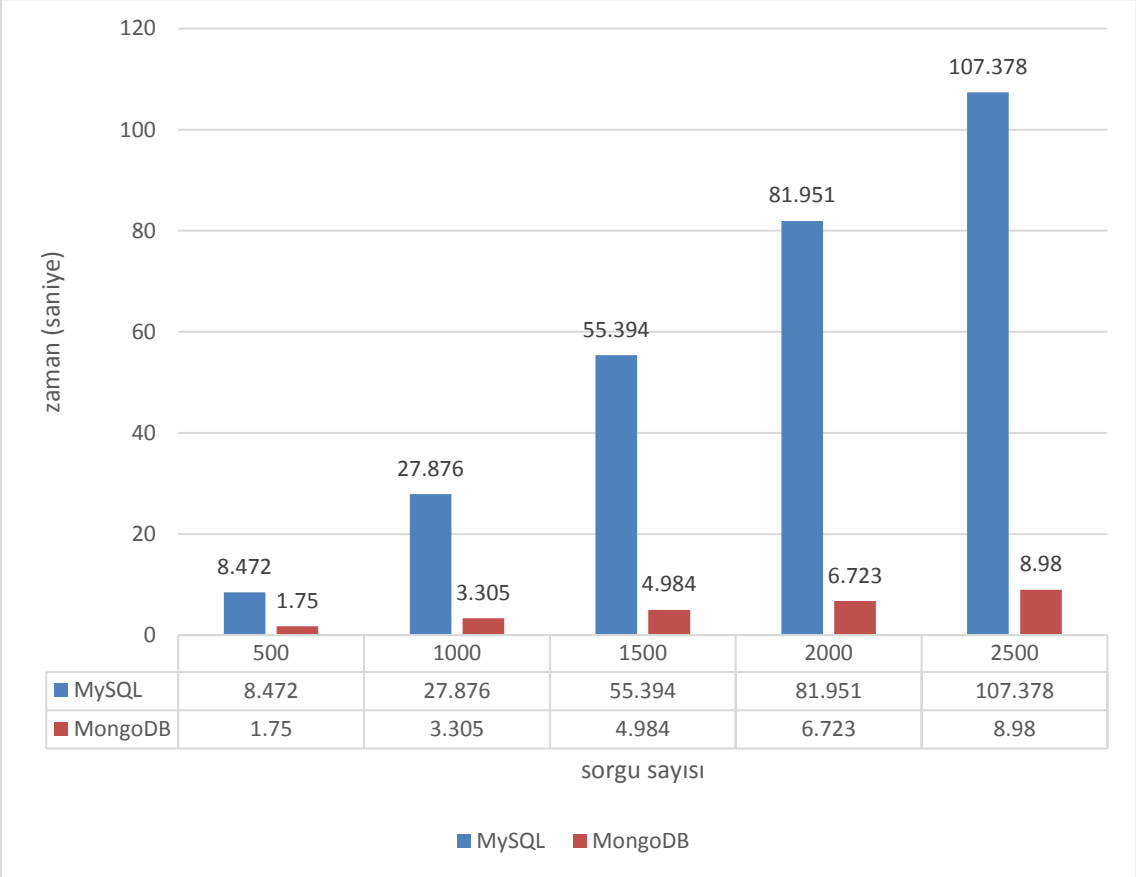
Şekil 4.13'te; MySQL ve MongoDB veri tabanlarına, üçüncü veri tabanı sorgusu olarak tanımlanan "INNER JOIN" terimi içermemekle birlikte iç içe geçmiş SELECT içeren karmaşık sorgu kodu ile uygulanan karşılaştırma testi sonuçları gösterilmiştir.



Şekil 4.13. Sorgu 3 kullanılarak veri tabanı sistemlerinin performans karşılaştırması

Şekil 4.13 ile ilgili yapılan analizlerde; INNER JOIN içermeyen bu karmaşık sorguda da sadece SELECT içeren basit sorguda olduğu gibi MySQL veri tabanı sisteminin MongoDB sistemine göre daha üstün bir performans gösterdiği gözlemlenmiştir. Bu veri tabanı sorgusu ile elde edilen test sonuçları, tablo birleştirmeye gerek olmadan yapılan karmaşık sorgulamalarda ilişkisel veri tabanlarının NoSQL veri tabanlarına göre %100'e yakın bir performans avantajı sağladığını göstermektedir.

Şekil 4.14'te; MySQL ve MongoDB veri tabanlarına dördüncü veri tabanı sorgusu olarak tanımlanan "INNER JOIN" terimi de içeren detaylı ve karmaşık sorgu kodu ile uygulanan karşılaştırma testi sonuçları gösterilmiştir.

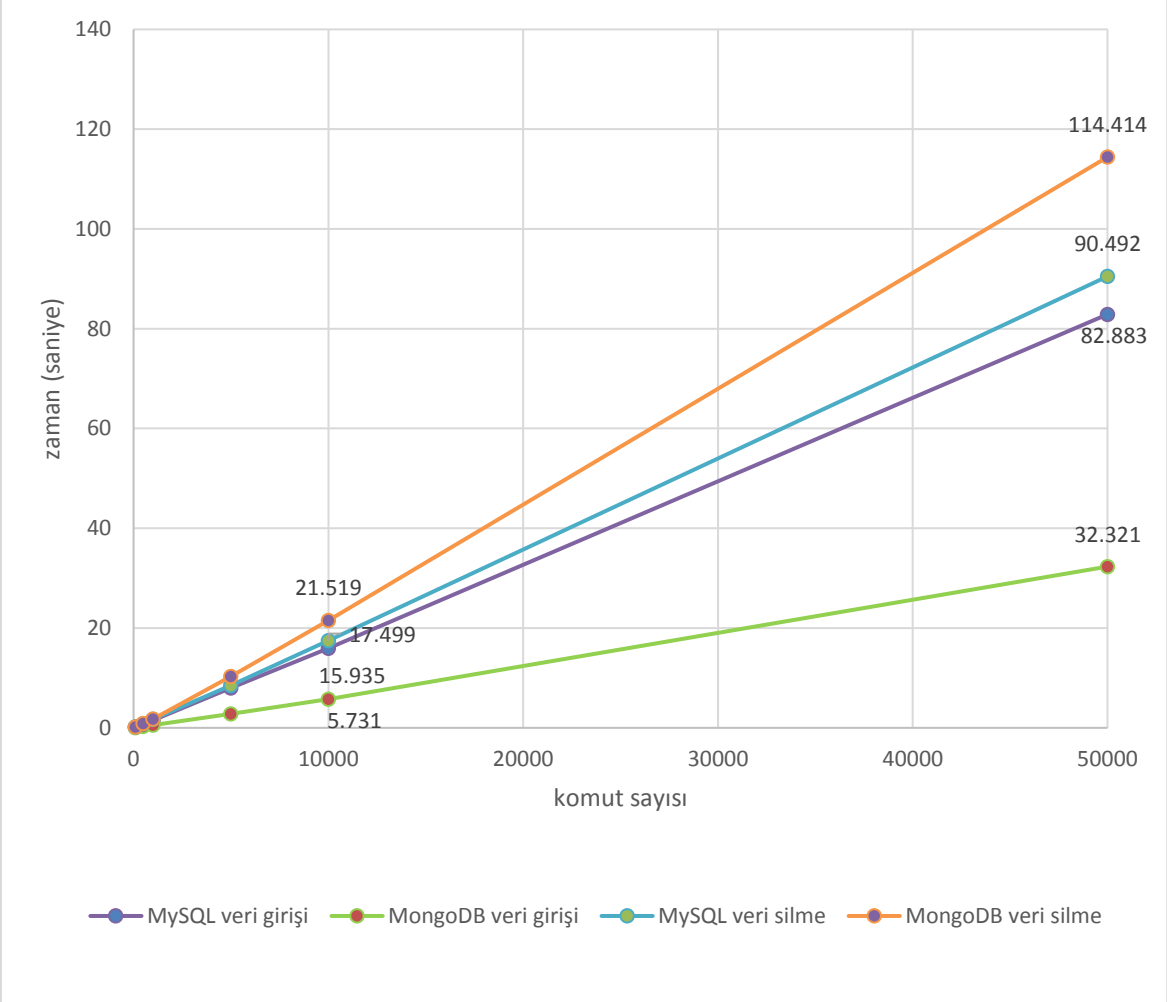


Şekil 4.14. Sorgu 4 kullanılarak veri tabanı sistemlerinin performans karşılaştırması

Şekil 4.14 ile ilgili yapılan analizde; tablo yerine koleksiyon yapısını kullanan MongoDB veri tabanı sisteminin MySQL veri tabanı sistemine kıyasla oldukça üstün bir performans gösterdiği gözlemlenmiştir. Bu karşılaştırma, eşdeğer veri kayıt setlerinde yapılmasına rağmen MongoDB veri tabanı sisteminin sorguları işleme ve sonuçlandırma süresinin oldukça düşük olduğu kolaylıkla görülebilmektedir. MongoDB bu şekildeki veri tabanı sorgulamalarında daha belirgin ve istikrarlı bir performans göstermiştir.

Bu sonuçlardan; MongoDB veri tabanı sisteminin, ilişkisel veri tabanlarında kullanılan veri tabanı şeması yerine alt şema belgelerini kullanarak az sayıdaki koleksiyonları sorguladığı yapısının, bu tarz karmaşık sorgularda ne denli zaman tasarrufu sağladığı gözlemlenmiştir.

Son olarak şekil 4.15'te; MySQL ve MongoDB veri tabanlarına uygulanan, elli bin kayda kadar veri ekleme "INSERT" ve silme "DELETE" işlemleri sonuçları gösterilmiştir. Veri tabanlarında, çoğunlukla veri okuma ve yazma işlemleri yapılmasına rağmen imkân dahilinde olduğu için bu çalışmada veri silme performansları da karşılaştırılmıştır.



Şekil 4.15. Veri tabanı sistemlerinin veri ekleme ve silme performansı kıyaslaması

Şekil 4.15 ile ilgili yapılan analizde; her iki veri tabanının komut sayılarına göre işlem sürelerinin doğrusal bir eğilim gösterdiği gözlemlenmiştir. MongoDB veri tabanı sisteminin veri ekleme işlemlerinde MySQL veri tabanı sistemine göre çok daha iyi bir performans ortaya koymuştur. Veri girişi işlemi elli bine kadar arttırıldığında bu performansın giderek daha da belirginleştiği ve yaklaşık üç kata kadar arttığı rahatlıkla gözlemlenebilmektedir. Veri silme işlemlerinde ise MySQL veri tabanı sistemi MongoDB sistemine oranla daha iyi bir performans sergilemiştir.

Veri silme işlemleri için öncelikle arama yapılarak ilgili kaydın bulunması ve sonrasında silme işleminin gerçekleştirilmesi gerektiği için veri ekleme işlemlerine oranla genel olarak daha uzun sürmüştür. Bu işlemlerde, MongoDB ve MySQL veri tabanı sistemlerinin komut sayısı az olduğunda benzer bir performans gösterdiği ancak sayı arttırıldığında MySQL sistemlerin MongoDB veri tabanı sistemlerine oranla daha iyi bir performans gösterdiği gözlemlenmiştir. Bu performans farkı elli bin kayıt sayısında %20 civarında gerçekleşmiştir.

MongoDB ve MySQL veri tabanı sistemlerinin veri ekleme ve veri silme performansının karşılaştırıldığı bu son testte, veri ekleme performans farkının sebebi MongoDB veri tabanı sisteminin operasyonel farklılığından kaynaklandığı değerlendirilmektedir. Veri silme testi sonucunda elde edilen MySQL performans farkı ise testin tutarlılığını göstermiştir. Çünkü, bir silme işlemi için veri tabanı seviyesinde doğru veri veya veri setini bulmak için arama işlemi yapılması gerekir. Bu, her iki veri tabanının “SELECT” ile benzer küçük bir arama işlemi gerçekleştirmesi anlamına gelir. Basit SELECT sorgularına daha iyi bir reaksiyon gösteren MySQL veri tabanı sisteminin, veri silme “DELETE” işlemlerinde de bu avantajını sürdürdüğü gözlemlenmiştir.

5. SONUÇ VE ÖNERİLER

Günümüzde meydana gelen teknolojik gelişim, veri tabanı yönetim sistemleri alanında da ciddi şekilde hissedilmiş ve bu alandaki gelişmeler NoSQL veri tabanı teknolojilerinin ortaya çıkmasına neden olmuştur. Yakın geçmişe kadar ilişkisel veri tabanı yönetim sistemlerini (RDBMS) tek alternatif olarak değerlendiren veri tabanı yöneticileri, yaşanan bu teknolojik gelişimler nedeni ile artık bu alanda geniş bir araştırma yapmak zorunda kalmıştır. Büyük veri ve dikey ölçeklenebilirlik maliyeti dezavantajlarının yarattığı ihtiyaca çözüm olarak geliştirilen NoSQL veri tabanı teknolojileri, mükemmel bir teknoloji olarak görülebildiği gibi, SQL sorgulama dilinden farklı bir dil kullanması veya tutarlılık gibi dezavantajları ile henüz tek başına yeterli olmadığı eleştirilerine de maruz kalmaktadır. Bu çalışmaya başlamadan önce, firma veya kurumların yönetim bilgi sistemlerinde geliştirilerek yöneticilerin karar vermesine destek olan uygulamalarda, hangi veri tabanı sisteminin kullanmasının verimli olabileceğinin ortaya çıkarılması gerekliliği belirlenmiştir.

Literatürde konu ile ilgili çalışmalar incelenmiş; daha çok yeni bir kavram olan NoSQL veri tabanı yapılarının özellikleri, kendi içinde mukayesesi ve büyük veri (big data) kapsamında kullanım avantajları ile ilgili çalışmalar olduğu gözlemlenmiştir. Bu çalışmalarda, ilişkisel olmayan (NoSQL) veri tabanlarının tutarlılıktan feragat edildiği taktirde büyük verinin ortaya çıkarttığı performans probleminin çözümünde etkili olabildiği ortaya çıkarılmıştır. Ancak konunun yönetim bilişim sistemleri kapsamında ele alındığı ve farklı veri tabanı sorgu türlerine göre ilişkisel ve NoSQL veri tabanı yönetim sistemlerinin nasıl reaksiyon gösterdiğinin ölçümlendiği dolayısıyla uygulamalarda hangi veri tabanı teknolojilerinin kullanılmasının daha verimli olabileceği ile ilgili bir çalışmaya rastlanmamıştır.

Bugün çoğu işletme mevcut uygulamalarında ilişkisel veri tabanı yönetim sistemlerini kullanmaktadır. Mevcut veri tabanı yönetim sistemlerinin, transactional olamayan NoSQL veri tabanı sistemlerine dönüştürülmesinin, tutarlılıktan feragat edilmesine neden olması, farklı sorgulama dili ve uygulamalarda bazı değişiklikler yapılmasını gerektirdiği için ilave iş yükü getirmesi gibi dezavantajları karşısında, hız avantajı yanı sıra sonradan yapılmak zorunda kalan değişikliklerde koleksiyon veri yapısı sayesinde esneklik sağlaması ve yatay olarak genişleyebileceği için maliyet avantajı sağlayacağı da tartışmasız bir gerçektir.

Bu çalışmada, ilişkisel ve NoSQL veri tabanı yönetim sistemlerinin performans karşılaştırmasının yapılması, yöneticilerin karar vermesine destek olmak üzere hazırlanan uygulamaların kullandığı veri tabanı sorgu çeşitlerine göre her bir veri tabanını nasıl reaksiyon gösterdiği ve dolayısıyla hangi teknolojinin hangi durumlarda diğerinden daha verimli olduğunun araştırılması amaçlanmıştır.

Karşılaştırma yapılabilmesi maksadıyla, geliştirilen uygulamalarda karşılaşılabilecek farklı veri tabanı sorgularının örneklendirilebileceği dört farklı veri tabanı sorgusu seçilmiş ve elde edilen sonuçlar, saniyedeki sorgu sayısı grafiklerinde gösterilerek işletmelere hangi durumda nasıl bir veri tabanı yönetim sistemi kullanmaları konusunda fikir verilmiştir.

Yapılan analizlerde; sadece “SELECT” komutu içeren basit sorgularda ve “INNER JOIN” içermeyen karmaşık sorgularda MySQL veri tabanı sisteminin daha iyi bir performans sergilediği gözlemlenmiştir. “INNER JOIN” içeren karmaşık ve detaylı karmaşık sorgularda ise, MongoDB veri tabanı sisteminin MySQL sistemine oranla bariz bir performans üstünlüğü sergilemiştir. NoSQL veri tabanı yönetim sistemlerinin koleksiyon veri yapısı ve alt belge kullanımı özelliklerinin, verileri farklı tablolarda tutarak belirli bir ilişki vasıtasıyla birbirine bağlayan ilişkisel veri tabanlarına oranla, tablo birleştirmek zorunda kalınan karmaşık sorgularda, çok büyük bir avantaj sağladığı gözlemlenmiştir. Tablo birleştirme gerektirmeyen sorgulamalarda MySQL veri tabanı sisteminin sağladığı performans, sorguların karmaşıklığına göre %20 ila %100 arasında değişirken, tablo birleştiren veri tabanı sorgularında MongoDB veri tabanı sistemlerinin sağladığı performans avantajı, sorguların karmaşıklığı ile doğru orantılı olarak 10 kata kadar çıkmıştır. Bu durum, çok sayıda tablo birleştirerek sonuç döndürülmesini sağlayan veri tabanı sorguları kullanılan uygulamalarda NoSQL sistemlerin kullanımını ön plana çıkartmıştır.

Yapılan son performans testleri ile veri tabanlarının veri ekleme ve silme performansları kıyaslanmıştır. Her iki veri tabanı, kayıt sayısı arttıkça doğrusal bir performans sergilemiştir. Veri ekleme işlemlerinde MongoDB veri tabanı sistemi daha iyi bir performans sergilerken, veri silme işlemlerinde MySQL daha üstün performans sağlamıştır. Veri silme işlemlerinde öncelikle verinin basit bir sorgu ile bulunması gerektiği için MySQL veri tabanı sisteminin bu performansı ilk testteki performansını teyit etmiştir.

Firmaların yönetim bilgi sistemlerince çok sık kullanılan uygulamalar baz alınarak seçilen bir uygulama ile, ilişkisel ve NoSQL veri tabanlarının performans karşılaştırması yapılan bu çalışmada, her iki veri tabanı modellerinin de üstün olduğu, avantaj ve dezavantaj sağladığı hususların mevcut olduğu rahatlıkla söylenebilmektedir. Firma veya kurumlarca yaygın olarak kullanılan ilişkisel veri tabanlarının sağladığı; SQL sorgulama dili, ACID işlevselliği ve bu kapsamda transactional olma avantajı yanı sıra "JOIN" içermeyen basit veri tabanı sorguları içeren uygulamalarda NoSQL veri tabanı sistemlerine oranla hız avantajı da sağladığı tespit edilmiştir. Yeni geliştirilen ve dünya üzerinde yaygın olarak kullanılmaya başlanan NoSQL veri tabanı sistemlerinin ise, ölçeklenebilirlik, alt belge kullanımı sağlayan farklı şema yapısı ile esneklik sağlaması gibi avantajlarının yanı sıra JOIN içeren karmaşık veri tabanı sorguları ile hazırlanan uygulamalarda çok ciddi bir performans avantajı sağladığı gözlemlenmiştir. Tablo birleştiren veri tabanı sorgulamalarında NoSQL veri tabanı yönetim sistemlerinin sağladığı performans avantajı, basit sorgulamalardaki ilişkisel veri tabanı yönetim sistemlerinin (RDBMS) sağladığı avantajdan göreceli olarak çok daha fazladır. Bu kapsamda firma veya kurumların yeni geliştirdiği uygulamalarda transaction gerektirmediği taktirde NoSQL veri tabanı yönetim sistemlerini kullanması faydalı olacaktır. Mevcut uygulamalarda ise getirdiği ilave iş yükü maliyetine rağmen uzun vadede sağlayacağı avantajlar göz önüne alınarak, transaction gerektiren ve sadece basit veri tabanı sorguları kullanan uygulanmalar dışında NoSQL veri tabanı yönetim sistemlerine geçilmesinin uygun olduğu değerlendirilmektedir.

Yapılan çalışma ile elde edilen sonuçlardan hareketle, artan veri tabanı yönetim sistemi alternatiflerinin, hazırlanan uygulamaların özelliklerine göre seçimlerini kolaylaştıracak bir çalışma yapılabileceği değerlendirilmektedir. Piyasada yaygın olarak kullanılan bütün veri tabanı yönetim sistemlerini tek tek değerlendirilerek geliştirilecek bir metot ile; hazırlanan uygulama kapsamında veri tabanına kaydedilecek tahmini kayıt sayısı, transactional özellikte işlem yapılması gerekip gerekmediği, kullanılacak veri tabanı sorgulamalarında tablo birleştirmenin gerektiği karmaşık sorguların kullanılıp kullanılmayacağı, zamanla yatay olarak ölçeklenebilirliğe ihtiyaç duyulup duyulmaması olasılığı gibi verileri kullanıcılardan alarak veri tabanı yöneticilerine kullanabilecekleri uygun veri tabanı yönetim sistemleri alternatiflerinin sunulacağı bir uygulama geliştirilebileceği öngörülmektedir.

KAYNAKLAR

1. Emhan, A. (2007). Karar Verme Süreci ve Bu Süreçte Bilişim Sistemlerinin Kullanılması. *Elektronik Sosyal Bilimler Dergisi*, 6 (21), 212-224.

2. İnternet: *Veri tabanı motorlarının kullanım sıralaması.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fdb-engines.com%2Fen%2Franking&date=2017-09-19, Son Erişim Tarihi: 19.09.2017

3. Otey, M. (2010). NoSQL? No Way! No fear that SQL Server will be replaced by non-relational databases. *SQL Server Magazine*, 5.

4. Orend, K. (2010). *Analysis and classification of nosql databases and evaluation of their ability to replace an object relational persistence layer,* Yüksek Lisans Tezi, Münih Üniversitesi, Münih, Almanya, 76-77.

5. Muhammad, Y. (2011). *Evaluation and ımplementation of distributed nosql database for mmo gaming environment,* Yüksek Lisans Tezi, Uppsala Üniversitesi, Upsala, İsveç, 45.

6. Hecth, R., Jablonski, S. (2011). *NoSQL evaluation a use case oriented survey,* International Conference on Cloud and Service Computing, 336-341.

7. Brad, C. N. (2012). *Data de-duplıcatıon ın NoSQL databases,* Yüksek Lisans Tezi, University of Saskatchewan, Saskatchewan, Kanada, 91-93.

8. Roijackers, J.A.M (2012). *Bridging SQL and NoSQL,* Yüksek Lisans Tezi, Eindhoven University of Technology, Eindhoven, Hollanda, 65-70.

9. Näsholm, P. (2012). *Extracting data from nosql databases a step towards ınteractive visual analysis of NoSQL data,* Yüksek Lisans Tezi, Chalmers University of Technology, Göteborg, İsveç, 57-61.

10. Shermin, M. (2013). *An access control model for nosql databases,* Yüksek Lisans Tezi, The University of Western Ontario, London, Ontario, Kanada, 65-67.

11. Benzer, U. (2014). *Ontoloji arama motorlarının incelenmesi ve NoSQL tabanlı bir ontoloji havuzu gerçekleştirimi,* Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir, 56-70.

12. Yar, T. Y., Tun, K. M. L. (2014). Data relationship query in relational db, NoSQL db and graph db. *International Journal of Computer Applications*, (105), 17, 31-37.

13. Uzunbayır, S. (2015). *Büyük veri tasarım zorlukları üzerine bir sosyal alışveriş uygulaması kullanılarak ilişkisel veri tabanları ve nosql akımları arasında bir karşılaştırma,* Yüksek Lisans Tezi, İzmir Ekonomi Üniversitesi, İzmir, 106-117.

14. Aladily, A. (2015). *En çok kullanılan nosql veri tabanları performans karşılaştırması,* Yüksek Lisans Tezi, Kadir Has Üniversitesi, İstanbul, 26-44.

15. Karataş, G. (2015). *Dağıtık veri tabanları için test uygulaması,* Yüksek Lisans Tezi, İstanbul Kültür Üniversitesi, İstanbul, 77-88.

16. Ayyagari, N. R. (2015). *Databases for mediation systems,* Yüksek Lisans Tezi, Blekinge Institute of Technology, Karlskrona, İsveç, 45-50.

17. Aydemir, F. (2016). *Sınır güvenliği için büyük veri teknik ve teknolojileri ile boru hattı tasarımı,* Yüksek Lisans Tezi, Gazi Üniversitesi, Ankara, 43-48.

18. Hammood, A. H. (2016). *NoSQL veri tabanı sistemlerinin karşılaştırılması: mongodb, apache hbase, and apache cassandra üzerine bir çalışma,* Yüksek Lisans Tezi, Çankaya Üniversitesi, Ankara, 40-50.

19. TAHA, İ. A. (2017). *NoSQL ve ilişkisel veri tabanı yönetim sistemlerinin kapsamlı karşılaştırılması,* Yüksek Lisans Tezi, Çankaya Üniversitesi, Ankara, 46-55.

20. Eren, K. (2017). *Bulut bilişim teknolojileri ve NoSQL veri tabanları kullanarak Türkiye'de terör olaylarının incelenmesi,* Yüksek Lisans Tezi, Sakarya Üniversitesi, Sakarya, 74-82.

21. Özatav, F. (2001). *Introduction to Information Technology* (First Edition). İstanbul: Bilgi Üniversitesi Yayınevi, 35.

22. Akman N. (2010). *Yönetim Bilişim Sistemleri Yüksek Lisans Dersi*, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, 6-7.

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

24. Karahoca, D., Karahoca, A. (1998). *İşletmeciler, Mühendisler ve Yöneticiler İçin Yönetim Bilişim Sistemleri* (Birinci Baskı). İstanbul: Beta Basım Yayım, 8,16-18

25. Hoşcan, Y. (2004). *Yönetim Bilgi Sistemi* (İkinci Baskı). Eskişehir: Anadolu Üniversitesi Yayınları, 23-25

26. Gökçen, H. (2002). *Yönetim Bilgi Sistemleri Analiz ve Tasarım Perspektifi*. Ankara: Epi Yayıncılık, 25.

27. Gümüştekin, G.E. (2004). İşletmelerde Yönetim Bilişim Sistemleri. *Yönetim ve Ekonomi*, Cilt: 11, Sayı: 1, 127.

28. Federico, A.P. (1985). *Management Information Systems and Organizational Behavior* (2nd edition). New York: Praeger Publishers Inc, 85.

29. Öğüt, A. (2012). *Bilgi Çağında Yönetim* (Beşinci Baskı). Ankara: Nobel Yayın Dağıtım, 84.

30. Karakaş, S. (1991). *Yönetim Enformasyon Sistemleri.* Kütüphane-Enformasyon-Arşiv Alanında Yeni Teknolojiler Sempozyumu Bildirileri (254-263). Yay (Haz. Hasan S. Keseroğlu). İstanbul: Türk Kütüphaneciler Derneği İstanbul Şubesi.

31. Özcan, B. (2006). *Yönetim Bilişim Sistemleriyle İşletmelerde Değer Yaratılması*, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı, İşletme Bilim Dalı, İstanbul, 41.

32. Lucey, T. (2014). *Management Information Systems* (Night Edition). Easleigh: DP Publications, 6.

33. Dirlik, S. (2008). *Yönetim Bilişim Sistemlerinin Örgüt Yapısına Ve Karar Vermeye Etkisi: Çoklu Örnek Olay Araştırması.* Yüksek Lisans Tezi, Muğla Üniversitesi, Sosyal Bilimler Enstitüsü, İşletme Anabilim Dalı, Muğla, 5.

34. Erbakırcı, A.M. (2008). *Ankara İli Ortaöğretim Okul Yöneticilerinin Teknolojiye Karşı Tutumları Ve Yönetim Bilişim Sistemlerini Kullanma Durumları,* Yüksek Lisans Tezi, Ankara Üniversitesi, Eğitim Bilimleri Enstitüsü, Eğitim Yönetimi Ve Politikası Anabilim Dalı, Eğitim Yönetimi Ve Teftişi, Ankara, 42.

35. Laudon, K. C., Laudon, J.P. (2015). *Essentials of Management Information Systems* (11th edition). Prentice Hall, 215-218.

36. Sarıhan, İ. H. (1998). *Teknoloji Yönetimi.* İstanbul: Desnet Yayınları, 202-203.

37. Gökçen, H. (2007). *Yönetim Bilgi Sistemleri.* Ankara: Palme Yayıncılık, 36.

38. Polat, N. (2006). Yönetim Bilgi Sistemi ve Sayıştay'da Yürütülen Çalışmalar. *Sayıştay Dergisi*, Sayı: 65, 192-193.

39. Yozgat, U. (1998). *Yönetim Bilişim Sistemleri.* İstanbul: Beta Yayınları, 170.

40. Dalay, İ. (2002). *Temel İşletmecilik Bilgileri*. Sakarya: Sakarya Kitabevi, 16.

41. Demircan, M.L., Moltay, A. (1997). *Bilgiyi Yönetmek*. İstanbul: Beta Basım Yayım, 59.

42. Internet: Baycan, E. *Veri Tabanı Yönetim Sistemleri*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.gencklavyeler.com%2Fforum%2Fattachment.php%3Faid%3D1135&date=2017-09-13, Son Erişim Tarihi: 13.09.2017.

43. Çoruh, M. (2017). *Veri tabanı: "Felsefesi, Tasarımı ve Yönetimi": MS-Access ve SQL Server Projeleri* (Birinci Baskı). PublishDrive, 70.

44. İnternet: *Database model (veri tabanı modelleri).* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fen.0wikipedia.org%2Findex.php%3Fq%3DaHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvRGF0YWJhc2VfbW9kZWw&date=2017-09-20, Son Erişim Tarihi: 20.09.2017

45. Önder, E. (2005). *Yönetim Bilişim Sistemleri Kapsamında Web Tabanlı İlişkisel Veri tabanı Yönetim Sistemleri ve Bir Uygulama*. Yüksek Lisans Tezi, İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü, İşletme Anabilim Dalı, İstanbul, 80-82.

46. Özaslan, Ö. (2011). *Web Tabanlı Jeodezik Amaçlı Mekânsal Veri Tabanı tasarımı ve Uygulamaları.* Doktora Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Jeodezi ve Fotogrametri Mühendisliği, Geomatik Mühendisliği, İstanbul, 23-27.

47. Codd, E.F. (1970). A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM*, 13 (6), 377-387.

48. Gündüz, D. (2005). Veri tabanlarına ve SQL'e Giriş. *Akademik Bilişim Konferansı*, Konya.

49. İnternet: *Nesne Yönelimli Veri Modelleri.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.webbilgileri.com%2FMakaleler%2F45-nesne-yonelimli-veri-modelleri.aspx&date=2017-09-13, Son Erişim Tarihi: 13.09.2017

50. Stephens, R., R-Plew, R. (2003). *Veri tabanları* (Birinci Basım) (çev. N. G. Küçükler). Alfa Basım Yayım Dağıtım Ltd. Şti. (Eserin orijinali 2001'de yayımlandı), 36-40, 69.

51. Özaslan, Ö. (2003). *Jeodezik Veri Tabanı Tasarımı ve WEB Tabanlı Yönetimi*. Yüksek Lisan Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 7-12.

52. İnternet: Topçuoğlu, H. *Veri tabanı tasarımında izlenecek yol haritası.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Ftasarlamag.com%2Fveri-tabani-tasariminda-izlenecek-yol-haritasi%2F&date=2017-09-15, Son Erişim Tarihi: 15.09.2017.

53. T.C. Millî Eğitim Bakanlığı. (2012). *Bilişim Teknolojileri, Veri Tabanı Tasarımı*. Ankara: 11-26.

54. İnternet: Özakar, H. (2014). *Veri tabanlarında normalizasyon.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fbeltslib.net%2Fveri-tabanlarinda-normalizasyon.html&date=2017-09-15, Son Erişim Tarihi: 15.09.2017.

55. İnternet: Şeker, Ş. E. (2009). *BCNF (Boyce Codd Normal Form).* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fbilgisayarkavramlari.sadievrenseker.com%2F2009%2F03%2F05%2Fbcnf-boyce-codd-normal-form%2F&date=2017-09-15, Son Erişim Tarihi: 15.09.2017.

56. Sürmeli, F., Erdoğan, M., Erdoğan, N., Banar, K., Kaya, E, Sevim, A. (2006). *Muhasebe Bilgi Sistemi.* Eskişehir: Anadolu Üniversitesi Yayını, 77-84.

57. Azimli, M. (2013). *Hiyerarşik ilişkiler ve NoSQL veri tabanları,* Güz Dönemi Ödevi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Sakarya, 6-9.

58. İnternet: *MySQL (temel özellikleri, tarihçe, yönetim araçları).* URL: http://www.webcitation.org/query?url=https%3A%2F%2Ftr.0wikipedia.org%2Findex.

php%3Fq%3DaHR0cHM6Ly90ci53aWtpcGVkaWEub3JnL3dpa2kvTXlTUUw&date=2017-09-18, Son Erişim Tarihi: 18.09.2017

59. İnternet: *MySQL nedir? Detayları ile MySQL veri tabanı.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.hostbulucu.com%2Fmysql-nedir-detaylariyla-mysql-veritabani%2F&date=2017-09-19, Son Erişim Tarihi: 19.09.2017

60. İnternet: *NoSQL a relational database management system.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.strozzi.it%2Fcgi-bin%2FCSA%2Ftw7%2FI%2Fen_US%2Fnosql%2FHome%2520Page&date=2017-09-19, Son Erişim Tarihi: 19.09.2017

61. İnternet: Davaz, S. (2013). *NoSQL nedir? Avantajları ve dezavantajları.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fkodcu.com%2F2014%2F03%2Fnosql-nedir-avantajlari-ve-dezavantajlari-hakkinda-bilgi%2F&date=2017-09-19, Son Erişim Tarihi: 19.09.2017.

62. İnternet: *NoSQL 2009 (cassandra conference).* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fblog.sym-link.com%2F2009%2F05%2F12%2Fnosql_2009.html&date=2017-09-19, Son Erişim Tarihi: 19.09.2017

63. Ward, M. (2013). NoSQL Database in the Cloud: MongoDB on AWS*. Amazon Web Services E-Journal*, 2.

64. İnternet: Takkar, V. (2015). *NoSQL veri tabanı türleri.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.vikramtakkar.com%2F2015%2F12%2Fnosql-types-of-nosql-database-part-2.html&date=2017-09-19, Son Erişim Tarihi: 19.09.2017.

65. Rozanski, N., Woods, E. (2012). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives* (2nd Edition). USA: Pearson Education Inc.,36.

66. Vaish, G. (2013). *Getting Started With NoSQL.* United Kingdom: Packt Publishing, 87-90.

67. İnternet: Freedman, M. (2017). *Zaman serisi verileri: Neden ilişkisel veri tabanı yönetim sisteleri yerine NoSQL sistemleri kullanırız?* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fblog.timescale.com%2Ftime-series-data-why-and-how-to-use-a-relational-database-instead-of-nosql-d0cd6975e87c&date=2017-09-20, Son Erişim Tarihi: 20.09.2017

68. İnternet: *Zaman serisi verilerinin saklanması için kullanılan veri tabanları.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.percona.com%2Fblog%2F2017%2F02%2F10%2Fpercona-blog-poll-database-engine-using-store-time-series-data%2F&date=2017-09-19, Son Erişim Tarihi: 19.09.2017

69. Chodorow, K. (2013). *MongoDB: The definitive guide* (Second Edition). USA: O'Reilly Media Publisher, 3-5.

70. İnternet: Akdoğan, H. (2013). *NoSQL kavramı ve MongoDB.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fkodcu.com%2F2013%2F04%2Fnosql-kavrami-ve-mongodb%2F&date=2017-09-20, Son Erişim Tarihi: 20.09.2017

71. İnternet: Yıldız, B. (2015). *MongoDB giriş.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fbilgehanyildiz.com%2F2015%2F04%2F13%2Fmongodb-giris%2F&date=2017-09-20, Son Erişim Tarihi: 20.09.2017

72. İnternet: Balcı, A. (2014). *NoSQL ve elastic search.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fbalciali.wordpress.com%2Ftag%2Fscale-out%2F&date=2017-09-18, Son Erişim Tarihi: 18.09.2017.

73. İnternet: Karaağaç, U. (2013). *Tansaction ve ACID kavramları.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fumutkaraagac.wordpress.com%2F2013%2F09%2F12%2Ftransaction-ve-acid-kavramlari%2F&date=2017-09-18, Son Erişim Tarihi: 18.09.2017.

74. Cattell, R. (2010). Scalable SQL and NoSQL Data Stores. *SIGMOD Record*, (39), 4, 12-27.

75. İnternet: Kise, A. (2016). *Veri tutarlılık modelleri ACID vs BASE.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.abdullahkise.com%2F2016%2F02%2Fnosql-dunyas-2-veri-tutarllk-modelleri.html&date=2017-09-18, Son Erişim Tarihi: 18.09.2017

76. Oracle Corporation. (2011). *An Oracle White Paper, Oracle NoSQL Database*. USA: 7.

77. İnternet: Pasqua, J. (2017). *Schema-on-read vs schema-on-write.* URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.marklogic.com%2Fblog%2Fschema-on-read-vs-schema-on-write%2F&date=2017-09-18, Son Erişim Tarihi: 18.09.2017

EKLER

EK-1. Test sonuçları

Çizelge 1.1. Sorgu 1 (sadece SELECT içeren basit sorgu) ortalama zaman çizelgesi

Sorgular	MySQL	MongoDB
500	0,520	0,543
1 000	0,850	1,075
1 500	1,375	1,650
2 000	1,650	2,188
2 500	2,210	2,728

Çizelge 1.2. Sorgu 2 (JOIN içeren karmaşık sorgu) ortalama zaman çizelgesi

Sorgular	MySQL	MongoDB
500	1,413	1,024
1 000	2,769	2,348
1 500	4,156	3,515
2 000	5,641	4,639
2 500	6,964	5,283

Çizelge 1.3. Sorgu 3 (JOIN içermeyen karmaşık sorgu) ortalama zaman çizelgesi

Sorgular	MySQL	MongoDB
500	0,472	0,845
1 000	0,876	1,707
1 500	1,394	2,710
2 000	1,951	3,658
2 500	2,378	4,578

Çizelge 1.4. Sorgu 4 (iç içe JOIN içeren detaylı karmaşık sorgu) ortalama zaman çizelgesi

Sorgular	MySQL	MongoDB
500	8,472	1,750
1 000	27,876	3,305
1 500	55,394	4,984
2 000	81,951	6,723
2 500	107,378	8,980

EK-1. (devam) Test Sonuçları

Çizelge 1.5 Veri ekleme (INSERT) ve silme (DELETE) işlemleri ortalama zaman çizelgesi

Kayıt sayısı	MySQL veri ekleme	MongoDB veri ekleme	MySQL veri silme	MongoDB veri silme
100	0,152	0,050	0,163	0,192
500	0,756	0,277	0,813	0,857
1 000	1,547	0,543	1,729	1,794
5 000	7,978	2,832	8,522	10,288
10 000	15,935	5,731	17,499	21,519
50 000	82,883	32,321	90,492	114,414

EK-2. Konfigürasyon

İlişkisel ve NoSQL veri tabanı yönetim sistemlerinin karşılaştırıldığı bu çalışmada MySQL ve MongoDB veri tabanı sistemleri kullanılmıştır. Kullanılan MySQL veri tabanı yönetim sistemine; “https://dev.mysql.com/downloads/cluster” adresinden, MongoDB veri tabanı yönetim sistemine ise; “https://www.mongodb.com/download-center#community” adresinden ulaşılabilmektedir. İlgili programlar, kullanılan işletim sistemine uygun versiyon seçilerek indirilmelidir. Kurulum aşamasında yapılacak işlemler, ilgili internet sitelerinde yer almaktadır.

Veri tabanı sorgulamalarının yapılması için “Shell script” komutları kullanılmıştır. Kullanılan script komutları aşağıda listelenmiştir;

```
#! / bin / bash cd $HOME/ tests
#Number of threads to run on JOBS=4
#This command should be run i n the command line and not in
#parallel to create the database
#. / harness–d 0–r 1 –t 1 –x 10000 –i 500 –p 1
eval ‘ for i in $ ( eval echo " { 1 . . $JOBS} " ) ; do echo $ i ; done |parallel –q –j "$JOBS" . /
harness –d 0 –r 1 –t 5 –x 10000 –i 500 –p 0 –o / data / s q l / multnodes / test 1 –1–{ }.out ’
eval ‘ for i in $ ( eval echo " { 1 . . $JOBS} " ) ; do echo $ i ; done |parallel –q –j "$JOBS" . /
harness –d 0 –r 1 –t 5 –x 10000 –i 1000 –p 0 –o / data / s q l / multnodes / test 1 –1–{ }.out’
eval ‘ for i in $ ( eval echo " { 1 . . $JOBS} " ) ; do echo $ i ; done |parallel –q –j "$JOBS" . /
harness –d 0 –r 1 –t 5 –x 10000 –i 1500 –p 0 –o / data / s q l / multnodes / test 1 –1–{ }.out’
eval ‘ for i in $ ( eval echo " { 1 . . $JOBS} " ) ; do echo $ i ; done |parallel –q –j "$JOBS" . /
harness –d 0 –r 1 –t 5 –x 10000 –i 2000 –p 0 –o / data / s q l / multnodes / test 1 –1–{ }.out
eval ‘ for i in $ ( eval echo " { 1 . . $JOBS} " ) ; do echo $ i ; done |parallel –q –j "$JOBS" . /
harness –d 0 –r 1 –t 5 –x 10000 –i 2500 –p 0 –o / data / s q l / multnodes / test 1 –1–{ }.out’
```

ÖZGEÇMİŞ



Kişisel Bilgiler

Soyadı, adı : ÖZTÜRK, Serdar

Uyruğu : T.C.

Doğum tarihi ve yeri : 18.07.1984, Hendek

Medeni hali : Evli

Telefon : 0 (532)200 51 84

E-posta : serdarozturktr@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniversitesi /Bilişim Enstitüsü	Devam Ediyor
Lisans	Kara Harp Okulu/Sistem Mühendisliği	2006
Lise	Atike Hanım Anadolu Lisesi	2002

İş Deneyimi

Yıl	Yer	Görev
2006-Halen	Kara Kuvvetleri Komutanlığı	BS Plan Subayı

Yabancı Dil

İngilizce

Yayınlar

- Atmaca, H. E., Öztürk, S. (2017). İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi. *Internatıonal Journal of Informatıcs Technologıes*, sayı: 10(2), 199.



GAZİLİ OLMAK AYRICALIKTIR.