İLİŞKİSEL VE İLİŞKİSEL OLMAYAN VERİ TABANI SİSTEMLERİ

ömer
körükmez

GİRİŞ

BİLİŞİM TEKNOLOJİLERİ VE BİLGİSAYAR SİSTEMLERİ HIZLA GELİŞTİĞİ İÇİN SÜREKLİ FARKLI ALANLARA YAYILARAK ORTAYA BİR VERİ KARMAŞIKLIĞI ÇIKARMIŞTIR BU VERİLERİN DOĞRU ŞEKİLDE TUTULMASI İÇİN FARKLI VERİ TABANI YÖNTEMLERİ GELİŞTİRİLMİŞTİR

1-İLİŞKİSEL VERİ TABANI

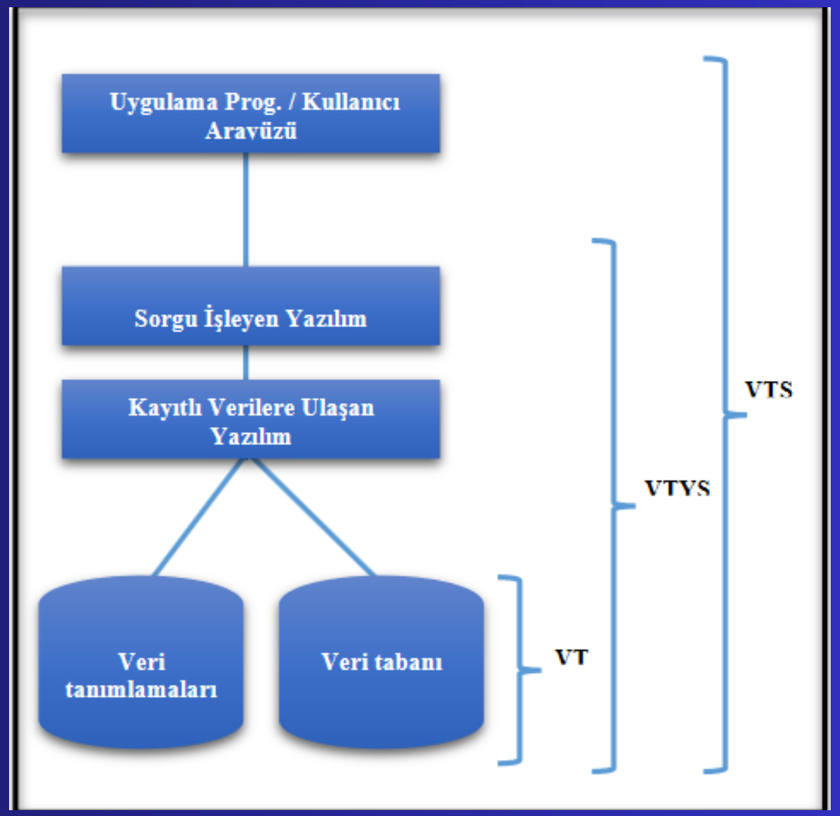
2- İLİŞKİSEL OLMAYAN VERİ TABANI

BİLŞİM SİSTEMLERİ VE YÖNETİMİ (INFORMATION SYSTEMS AND MANAGEMENT)

Bilişim sistemleri, bilgiyi toplama, düzenleme, işleme ve saklama süreçlerini içerir. Bu süreçler girdi, işlem ve çıktı aktiviteleriyle gerçekleşir. Girdi, ham bilgilerin toplanması; işlem, bu bilgilerin anlamlı biçimde dönüştürülmesi; çıktı ise işlenmiş bilginin kullanıcılara veya aktivitelere aktarılmasıdır. İşletmeler için bilişim sistemleri, yönetim çözümleri sunar ve etkin bir şekilde kullanabilmek için yönetim ve teknoloji konusunda bilgi sahibi olmayı gerektirir.

VERİ TABANI VE VERİ TABANI YÖNETİM SİSTEMLERİ (DATABASE AND DATABASEMANAGEMENT SYSTEM)

VERİ TABANI GENEL MANADA KULLANIM AMACINA UYGUN OLARAK İSTİFLENMİŞ VERİ TOPLULUĞUDUR VERİ TABANI YÖNETİM SİSTEMLERİ İSE VERİLERE AYNI ZAMANDA BİRDEN COK BAĞLANTI SAĞLAYABİLMEK VE VERİ TABANININ YÖNETİMİNİ SAĞLAMAK İLE İLGİLENİR



VERİ TABANI MODELLERİ

DÜZ MODEL VEYA TABLO MODELİ

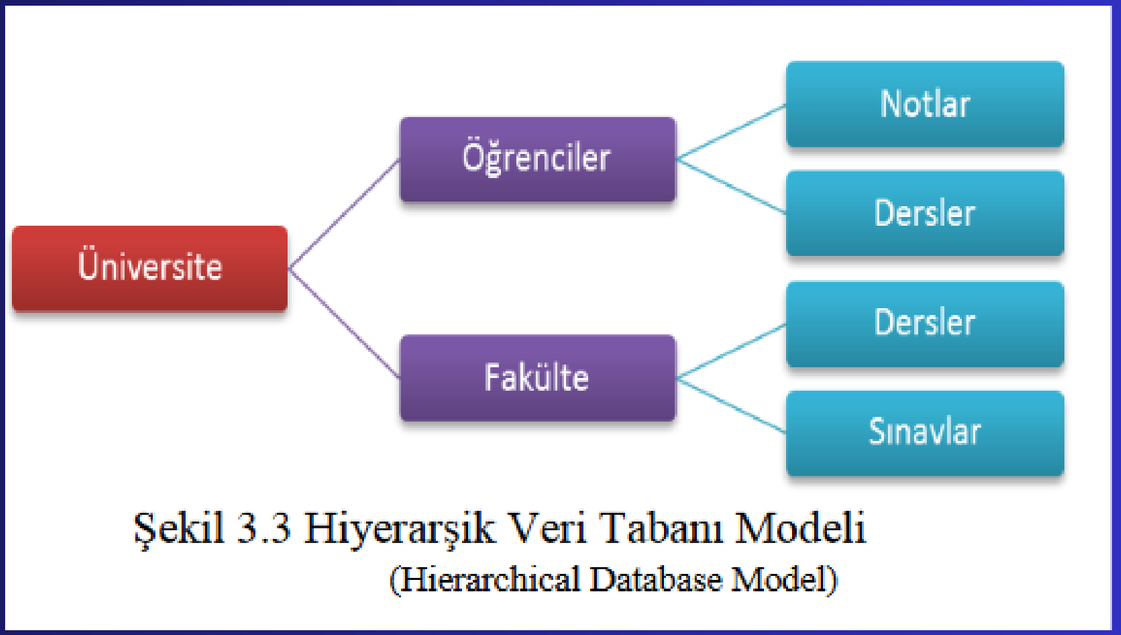
verilerin benzer özelliklerine göre sütunlarda gruplandığı ve veri gruplarının ise satırlarda yer aldığı iki boyutlu bir veri modelidir. Örneğin, kullanıcı adları ve şifrelerinin tutulduğu veri tabanı bu modele örnek olarak verilebilir. Her satırda bir kullanıcıya ait şifre bilgileri gibi benzer veriler bulunur ve sütunlarda ise aynı türde veriler yer alır. Bu model, tek bir tablodan oluşan bir veri modelidir

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

Şekil 3.2 Düz veri modeli örneği
(Instance of flat data model)

HİYERAŞİK VERİ MODELİ:

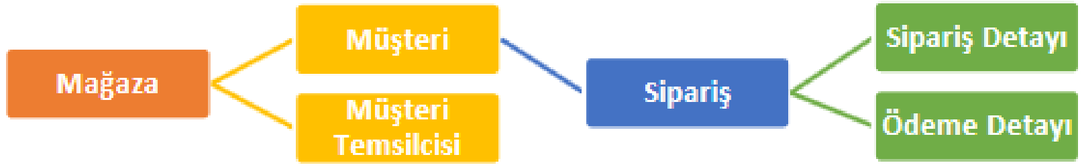
ADINI VERİ DEPOLAMA YÖNTEMLERİNDEN ALAN BU VERİ TABANI MODELİNİN YAPISAL VERİLERİNE KAYIT ADI VERİLİR BİR AĞAÇ GİBİDİR KÖK VE ÇOCUKLARDAN OLUŞUR



Şekil 3.3 Hiyerarşik Veri Tabanı Modeli
(Hierarchical Database Model)

AĞ VERİ MODELİ:

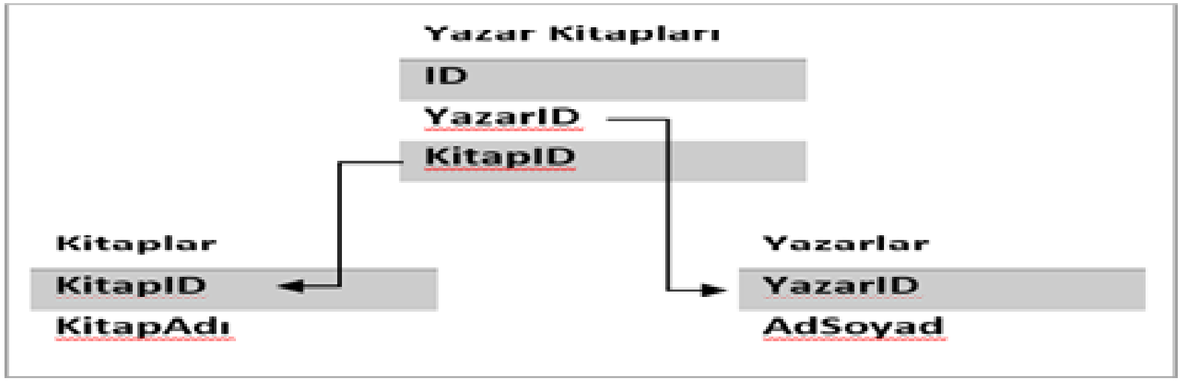
HİYERAŞİK VERİ MODELİNİN GELİŞMİŞ HALİDİR VERİLER BİRİBİRİ İLE İLİŞKİLİ OLDUGUNDAN KOLAY KABUL EDİLMİŞTİR BİRDEN ÇOĞA VEYA ÇOKTAN ÇOĞA İLİŞKİLER İÇEREBİLİR



Şekil 3.4 Ağ Veri Modeli
(Network Data Model)

İLİŞKİSEL VERİ MODELİ:

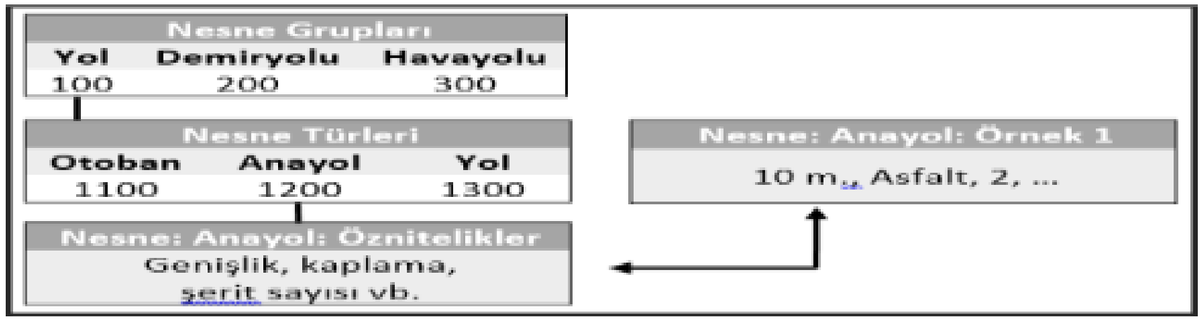
HİYERAŞİK VE AĞ VERİ MODELLERİNİN YETERSİZ KALMASI SONUCU ORTAYA CIKMIŞTIRTEMEL OLARAK İLİŞKİ KAVRAMINA DAYANIR ÇEŞİTLİ İLİŞKİ ÖRNEKLERİNDE OLUŞUR İKİ BOYUTLU SATIR VE SUTUNLARLA GÖSTERİLİR HER SATIRDA BİRBİRİİ İLE İLİŞKİLİ VERİLER DEPOLANIR



Şekil 3.5 İlişkisel Veri Modeli
(Relational Data Model)

NESNE YÖNELİMLİ VERİ MODELİ:

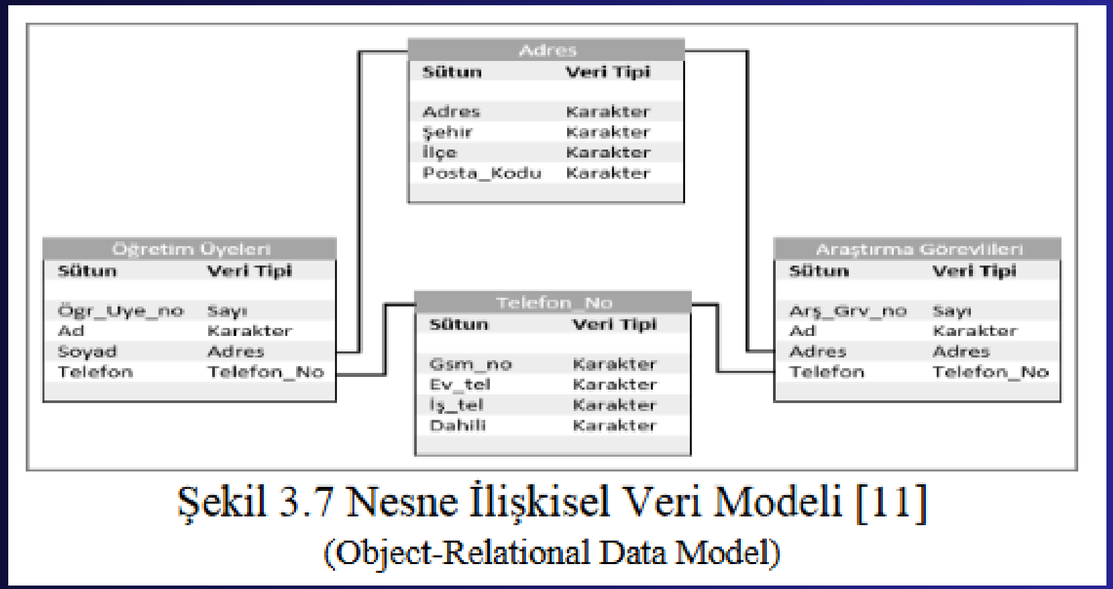
NESNE YÖNELİMLİ PROGRAMLAMAYA DAYANAN MODELDİR



Şekil 3.6 Nesne Yönelimli Veri Modeli
(Object-Oriented Data Model)

NESNE İLİŞKİSEL VERİ MODELİ:

Nesne ilişkisel veri tabanı, ilişkisel işlevselliğin üzerine nesne yönelimli özellikler içerir. İlişkisel veri tabanları içinde nesne yönelimli karakteristikler içeren ilk veri tabanı 1997 yılında piyasaya sunulan Oracle8'dir



Şekil 3.7 Nesne İlişkisel Veri Modeli [11]
(Object-Relational Data Model)

ÇOKLU ORTAM VERİ MODELİ:

NESNE İLİŞKİSEL VERİ TABANINA BENZER FİLİM,MÜZİK ,VİDEO GİBİ VERİLERİ İŞLEMEK İŞLERKEN ADIMLARI KULLANICIYA GÖSTERMEK AMACI İLE KULLANILIR GÖRÜNTÜLÜ EĞTİM ÜÇ BOYUTLU GÖRÜNTÜLEME GİBİ BİR ÇOK ALANDA KULLANILMAKTADIR

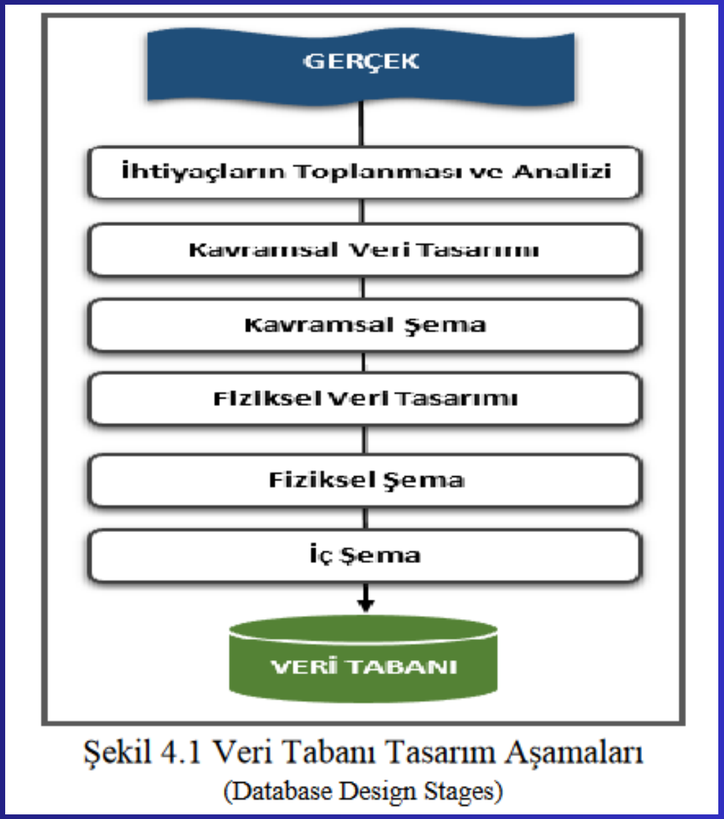
DAĞTIK VERİ MODELİ:

İKİ YADA DAHA ÇOK BİLGİSAYARDAKİ DEPOLANAN BİR AĞ ÜZERİNDEKİ VERİLERİ İŞLEMEK İÇİN KULLANILAN VERİ TABANI GURUBUDUR PARÇALANARAK SORGULAMA HIZLANDIRILIR BİRDEN COK VERİ TABANINDA ÇALIŞSADA TEK BİR VERİ TABANI KULLANIYORMUŞ GİBİ GÖRÜNÜR

VERİ TABANI TASARIMI (DATABASE DESIGN)

BU TASARIMDA GERÇEĞİN GEREKSİNİMLER ÇERÇEVESİNDE VERİ TABANINA AKTARILMASI AMAÇLANIR ÖNCE GEREKSİNİMLER BELİRLENMELİ VE SONRA VERİNİN MODELİNE GÖRE KULLANILACAK VERİ YAPISI BELİRLENMELİ KULLANICI VE BİLGİSAYAR TARAFINDAN ANLAŞILABİLECEK ŞEKİLDE TASARLANMALI BU TASARLAMAYA ŞEMA ADI VERİLİR KULLANICI VE BİLGİSAYARLAR İÇİN FARKLI DÜZENLER OLDUGUNDAN DOLAYI HER İKİ DÜZEY İÇİNDE FAKLI VERİ MODELLERİ YAPILMALIDIR

Geleneksel veri tabanı tasarımında kavramsal tasarım, gereksinimlere göre kavramsal şemanın belirlendiği aşamadır. Kavramsal şema, veri tabanının genel yapısını tanımlayarak kullanıcıların veri tabanını anlamalarını sağlar. Fiziksel tasarım aşamasında ise verinin en yüksek verim için fiziksel olarak nasıl organize edileceği belirlenir ve sonuç iç şema olarak adlandırılır. İç şema, depolama yapıları, kayıt formatları, veri tabanına giriş yolları ve diğer detayları tanımlayan fiziksel veri modellerini kullanır. Bu nedenle, iç şema, yazılım ve donanıma bağımlı bir tanımlamadır. Bu süreçte, kavramsal şemanın veri tabanı yönetim sisteminin veri modeline dönüştürülmesi gerekebilir



Şekil 4.1 Veri Tabanı Tasarım Aşamaları
(Database Design Stages)

İLİŞKİSEL VE İLİŞKİSEL OLMAYAN (NoSQL) VERİ TABANI SİSTEMLERİ (RELATIONAL AND NON-RELATIONAL DATABASE (NoSQL) SYSTEMS)

YAYGIN KULLANILAN BİR VERİ TABANI SİSTEMDİR SATIR VE SÜTÜNLARDAN OLUŞUR BU SATIR VE SÜTÜNLAR TABLOLARI OLUŞTURUR TABLOLAR BİRİBİRİ İLE İLİŞKİLİDİRLER BU İLİŞKİDEN SÖZ EDİLEBİLMESİ İÇİN EN AZ İKİ TABLONUN YER ALMASI VE VERİLERİN BİRBİRİ İLE İLİŞKİLENDİRİLİYO OLMASI LAZIM

ACID; klasik ilişkisel veri tabanı sistemlerinde sağlanan temel özellikler aşağıda sıralanmıştır

- **Bölünmezlik (Atomicity)**
- **Tutarlılık (Consistency)**
- **İzolasyon (Isolation)**
- **Dayanıklılık (Durability)**

İlişkisel olmayan (NoSQL) veri tabanı 1998 yılında ilk olarak Carlo Strozzi tarafından kullanılmıştır

NoSQL ilişkisel veri tabanı sistemlerine rakip olarak ortaya çıkmıştır

Günlük 7 TB`lık işlem hacmine sahip Twitter ve 10 TB`lık Facebook örneğindeki gibi, çok büyük verilerin depolanması ve yazılmasında ilişkisel veri tabanlarının eksik kaldığı hususlarda, yatay ölçekleme yapan dağıtık NoSQL çözümleri geliştirilmiştir.

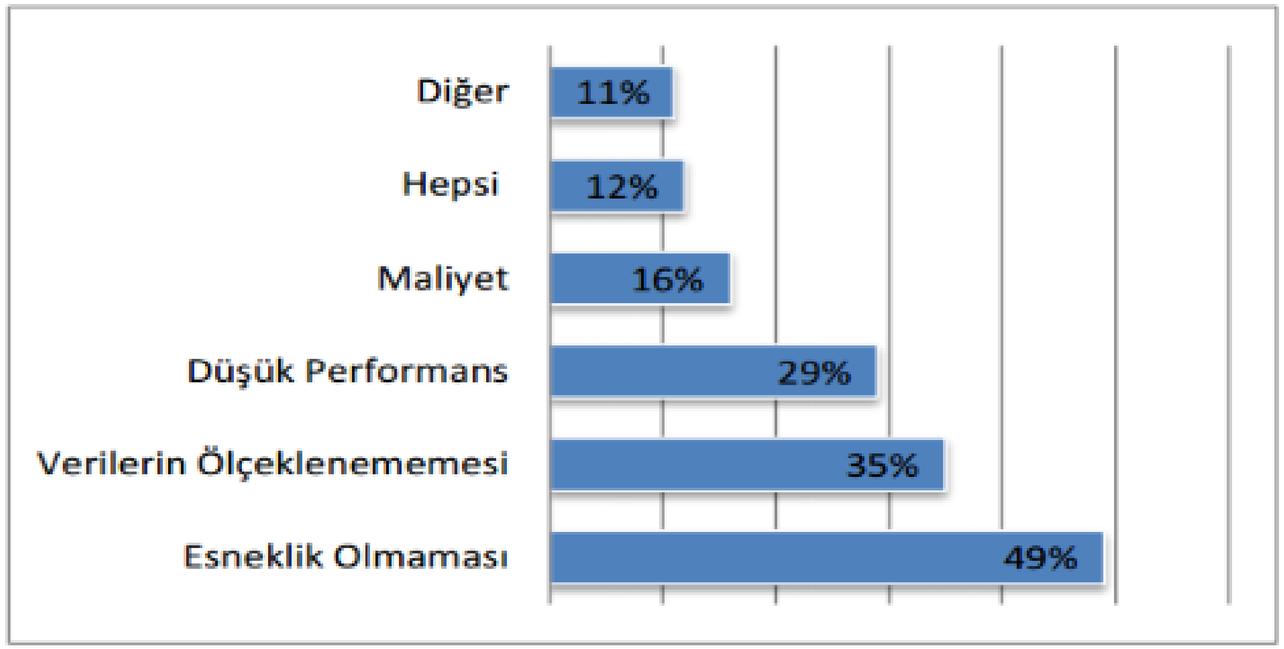
İlişkisel veri tabanı kullanıcılarının, araştırmalar neticesinde NoSQL veri tabanına geçmek istemelerinin nedenleri aşşağıda gösterilmiştir

İlişkisel veri tabanını yerine NoSQL veri tabanını tercihi gereksiz maliyeti azaltır

İlişkisel veri tabanlarının kullandığı ACID işlemselliğine

karşın NoSQL “BASE” (Basically Available- Soft state-

Eventually consistent) kısaltması ile ifade edilir



Şekil 5.1 Neden NoSQL Gerekli

(Why NoSQL Necessary)

Kolay Ulaşılabilirlik (Basically Available):veri ulaşım sorunlarını ortadan kaldırmak için bölünmemiş veriyi farklı sunuculardan sağlar

Esnek Durum (Soft state): ACID mantığından farklı olaran NoSQL tutarsız verileri kayıt edebilir

Eninde sonunda Tutarlı (Eventually consistent):uygulamalar anlık tutarlı olmasa bilene gelecek bi tarihte tutarlı olabilecegi savunulur

NoSQL veri yapısı henüz yeni olmasına karşın kendini kanıtlamış bir veri tabanı sistemidir

Tablo 1: Lider NoSQL ürünlerinin teknik karşılaştırması
(The Technical comparison about the leader NoSQL's products)

	Depolama	Protokol	Model	Lisans	Dil
MongoDB	Memory mapped b-trees	BSON	Document	AGPL	C++
CouchDB	COW-BTree	HTTP/REST	Document	Apache	Erlang
Riak	Pluggable: InnoDB, LevelDB, Bitcask	HTTP/REST or TCP/Proto	Key/Value	Apache	Erlang
Redis	In memory, snapshot to disk	bufs TCP	Key/Value	BSD	C++
Voldemort	Pluggable: BSV, MySQL, in-		Key/Value	Apache	Java
Cassandra	Memtable/SStable	TCP/Thrift	Wide Column	Apache	Java
HBase	HDFS	HTTP/REST or	Wide Column	Apache	Java

Sadeleştirme	Arama	Esinlenilen
Evet	Evet	
Hayır	Hayır	Dynamo
Evet	Evet	Dynamo
Hayır	Hayır	
Hayır	Hayır	Dynamo
Evet	Evet	BigTable, Dynamo
Evet	Evet	BigTable

VERİ TABANI MİMARİLERİNİN PERFORMANS KARŞILAŞTIRMASI (PERFORMANCE COMPARISONOFDATABASE ARCHITECTURE)

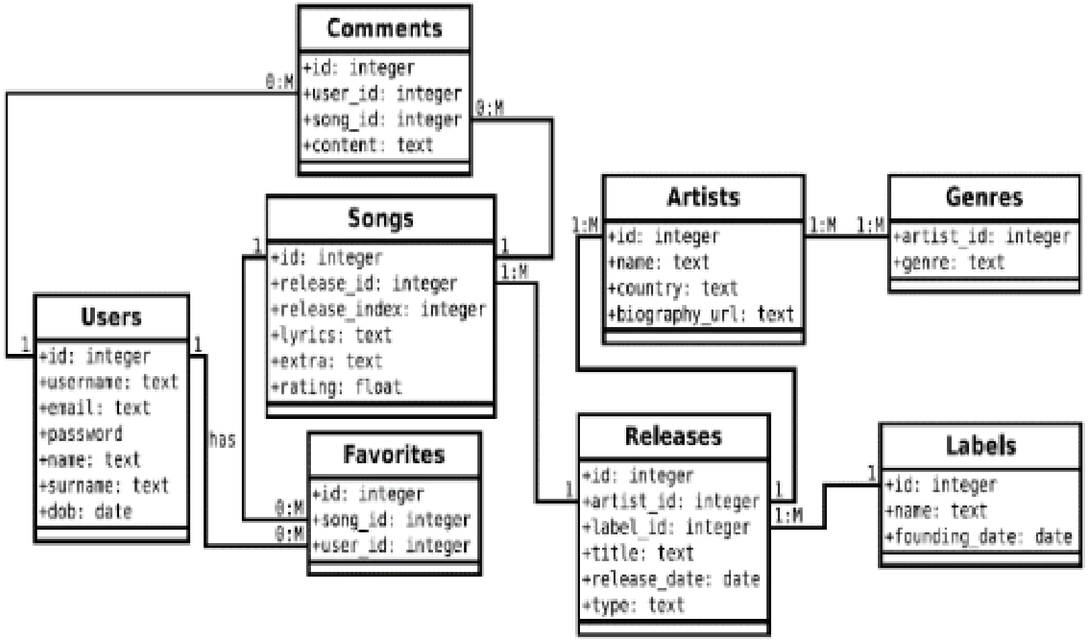
veri tabanı mimarilerinde oldukca bol çeşit vardır burada en yaygın kullanılan MySQL ve MongoDB veri tabanı sistemlerinin performans ve yatay ölçeklenebilirlik incelemesi için aşşağdaki madeller uygulanmalıdır

Veri tabanı sunucu sistemleri özellikleri belirlenmesi

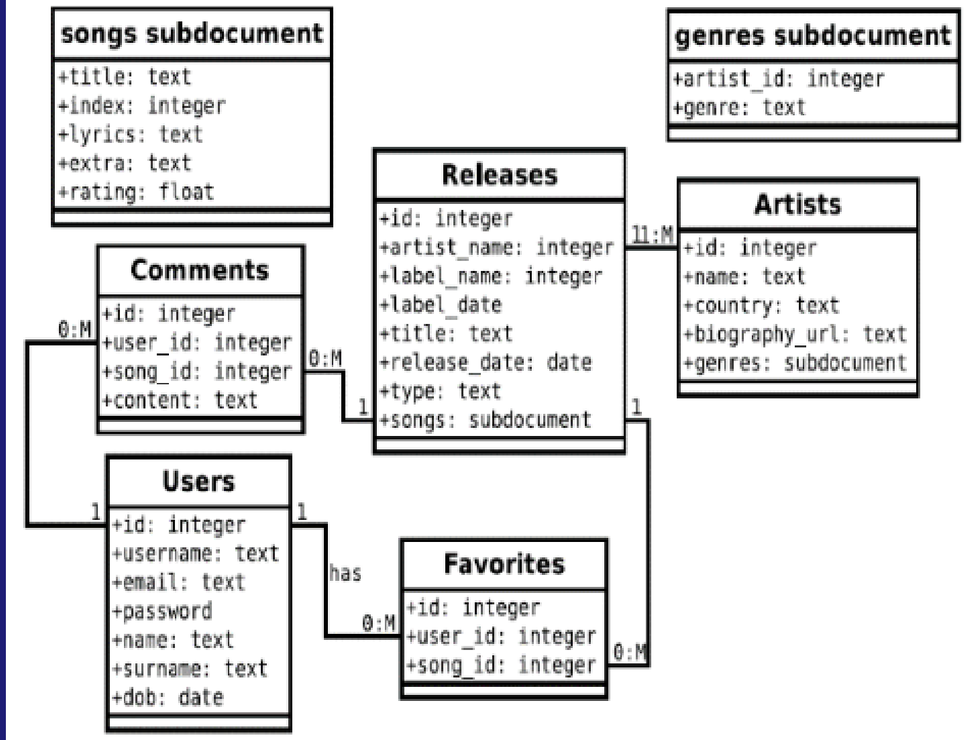
- **Veri tabanı şemaları oluşturulması**
- **Sorguların belirlenmesi**
- **Veri tabanı ayarlarının yapılması**
- **Ölçümler ve ölçüm metrikleri bilgileri**
- **Performans analizi ve sonuçlarıdır**

veri tabanı şemaları

kullanıcılara şarkı önermek için tasarlanmış MySQL ve MongoDB şemalarının karşılaştırılması aşşağıda verilmiştir



Şekil 6.1 MySQL veri tabanı şeması
(Relational database schema)



Şekil 6.2 MongoDB şeması
(MongoDB schema)

Veri Tabanı Sorguları:

yukarıdaki şarkı algoritması çaalışmasına ait aşşağıda üç farklı veri tabanı sorgusu kullanılmıştır

1-->“SELECT” içeren sorgu

Sorgu 1: Basit

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

2-->“INNER JOIN” içeren daha karmaşık bir sorgu

Sorgu 2: Karmaşık

```
SELECT ' Favourites. song_id ' AS fSID , ' Favourites. user_id ' AS fUID
FROM Favourites AS b INNER JOIN Favourites AS a
ON b. user_id = a. user_id
WHERE a. song_id = 123456 AND a. user_i d != 987654
```

3-->“SELECT” “JOIN”, “INNER JOIN” ve “WHERE” içeren sorgu

```
SELECT ' Songs. release_id ' AS sId , ' Releases. id ' ASrId
FROM Songs INNER JOIN Releases
ON Songs. release_id = Releases. id
WHERE artist_id IN
SELECT ' Genres. artist_id ' AS gAID
FROM Genres AS c
INNER JOIN Artists AS d
ON c.artist_id = d.id WHERE d.name = ' artist_name '
```

Ölçümler:

ölçümler için zaman çok önemlidir zaman ölçümleri ile ilgilde 3 farklı yöntem vardır

Birinci yöntem;
Clock() fonksiyonu kullanımı ile belirlibir süre CPU üzerinde harcanan zaman sonuçlarının eldedilmesini sağlamaktır

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

Üçüncü yöntem;
Slow Query Log (Yavaş sorgu kaydı)olarak adlandırılmaktadır.

Ölçüm Metrikleri:

Veri tabanlarının performansını ölçmek için ortak bir özellik gereklidir.Bir uygulama için en önemli faktör, bir görevi tamamlamak için gereken süre ve veri tabanının bir işlemi tamamlaması durumu için gerekli zamandır.

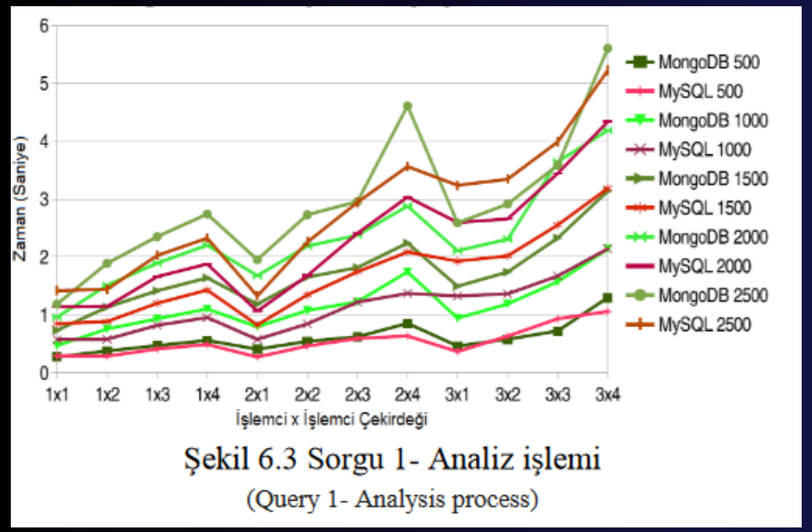
bunu hesaplamak için aşşağıdaki formül kullanılır

$$\text{Saniyedeki Sorgu} = \frac{\text{Toplam Sorgu Sayısı x Toplam İş Parçacığı Sayısı}}{\text{Ortalama Sorgu Süresi}}$$

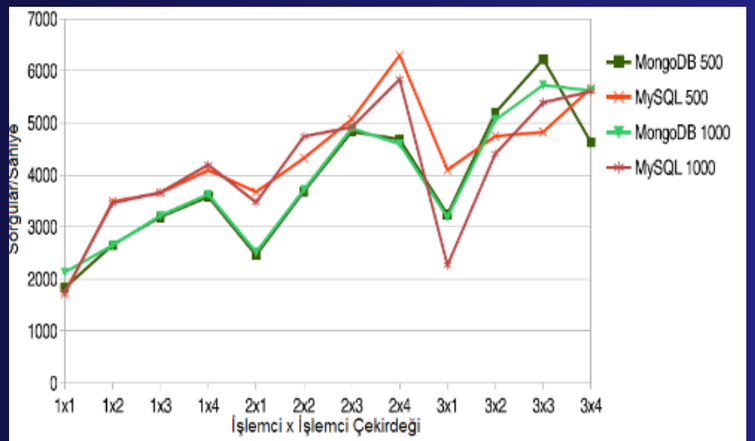
analiz ve sonuçlar

Sorgu Performansı Karşılaştırması:

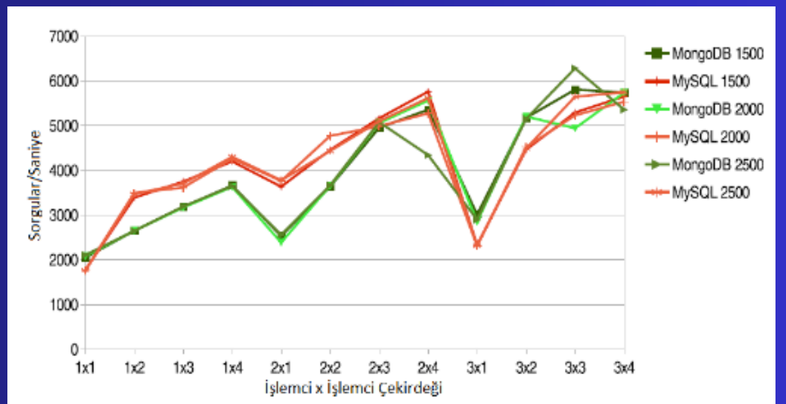
1-Sorgu sayısı arttıkça MongoDB'nin performansı düşüyor.
2-İşlemci çekirdeği sayıları değişmediğinde, performanslarında belirgin bir değişim olmuyor.
3-Özellikle MongoDB'nin belirli yapılandırmalarda daha avantajlı olduğu görülüyor.



Şekil 6.3 Sorgu 1- Analiz işlemi
(Query 1- Analysis process)



Şekil 6.4 Sorgu 1 - Sorgu/saniye analiz işlemi
(Query 1 - Queries/second analysis process)

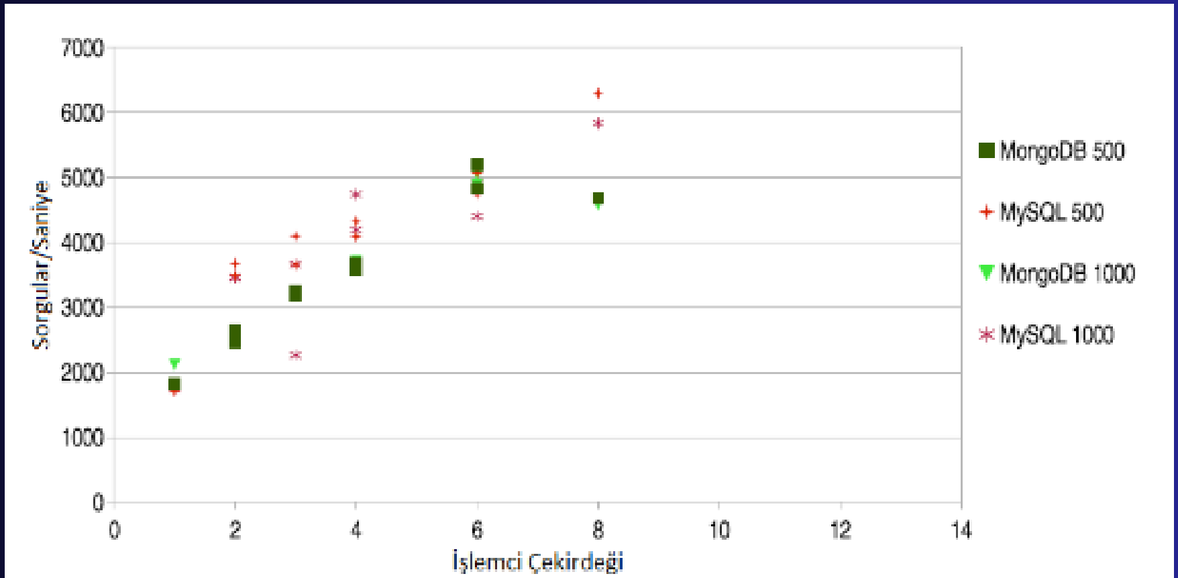


Şekil 6.5 Sorgu 1-Çok sayıdaki sorgu miktarı analiz işlemi
(Query 1- Many query amount analysis process)

İşlemci ve İşlemci Çekirdeği Yapılandırmalarının Etkisi:

1-İşlemci ve işlemci çekirdeği yapılandırmaları değiştikçe performans farklılıkları daha belirgin hale geliyor.

2-MongoDB, belirli yapılandırmalarda dahaavantajlı bir performans sergiliyor.



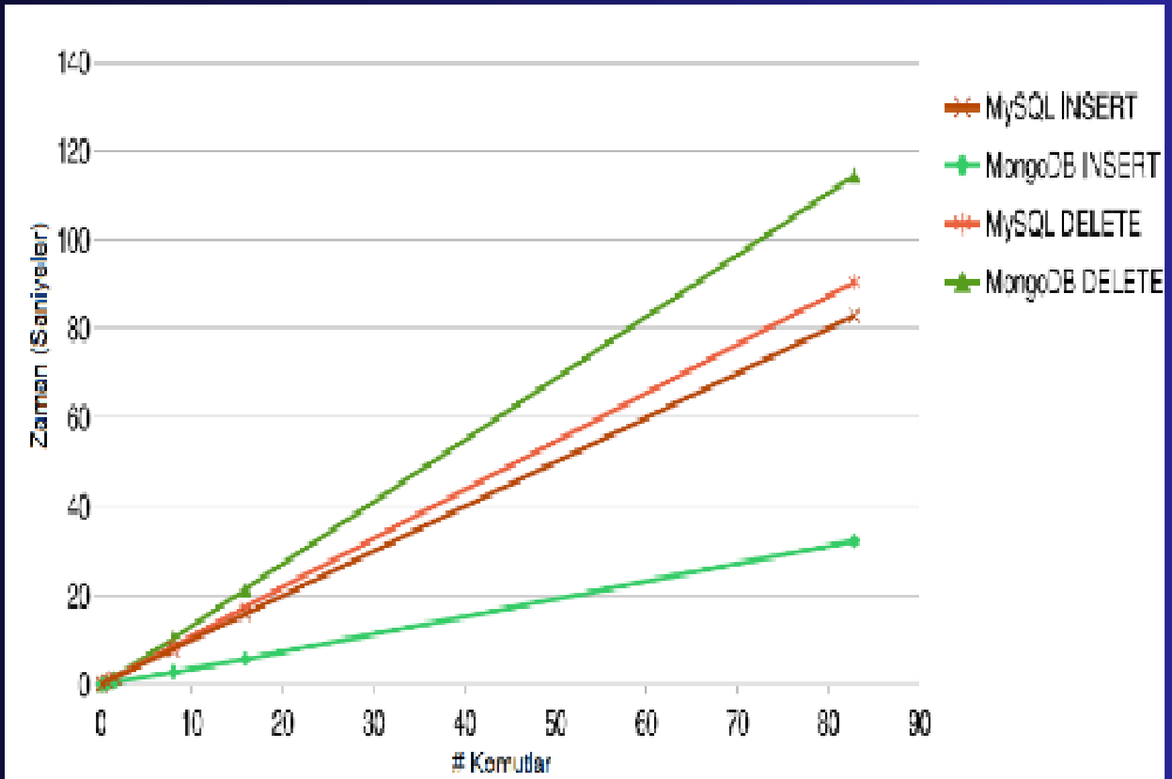
Şekil 6.6 Sorgu 1-Sorgular/Saniye ile işlemci çekirdeği miktarı için analiz işlemi

(Queries/second and amount of processor core analysis process)

Veri Ekleme ve Silme Performansı:

1-MongoDB, genellikle veri ekleme işlemlerinde MySQL'den daha iyi performans gösteriyor

2-Veri silme işlemlerinde ise benzer performans sergiliyor, ancak MySQL bazı durumlarda daha iyi performans gösterebiliyor.



Şekil 6.15 INSERT ve DELETE işlemleri

(INSERT and DELETE operations)

SONUÇ VE DEĞERLENDiRME (RESULT ANDEVALUATION)

bu çalışmada ilişkisel veri tabanları ile NoSQL veri tabanları karşılaştırılmış ve yönetim bilişim sistemleri açısından incelenmiştir. NoSQL veri tabanlarının büyük veri çiftleri, veri çoğaltma ve karmaşık sorgular konusunda avantajları olduğu belirlenmiştir. MongoDB'un özellikle eklemeler sırasında daha iyi bir performans sergilediği görülmüştür. İşletmelere hangi durumda hangi veri tabanı yönetim sisteminin daha uygun olduğu konusunda bilgi verilmiştir.