T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İLİŞKİSEL VE NOSQL VERİ TABANI SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

YÜKSEK LİSANS TEZİ

Caner SEÇGİN

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Temmuz, 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İLİŞKİSEL VE NOSQL VERİ TABANI SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

YÜKSEK LİSANS TEZİ

Caner SEÇGİN
(Y1513.010008)

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Metin ZONTUL

Temmuz, 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1513.010008** numaralı öğrencisi **Caner SEÇGİN**' in **"İLİŞKİSEL VE NoSQL VERİ TABANI SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 04.07.2018 tarih ve 2018/12 sayılı kararıyla oluşturulan jüri tarafından oybirliğiyle ile Tezli Yüksek Lisans tezi olarak ..kabul..edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**
Tez Savunma Tarihi : 20/07/2018	
1)Tez Danışmanı: Doç. Dr. Metin ZONTUL	.....
2) Jüri Üyesi : Prof. Dr. Ali GÜNEŞ	.....
3) Jüri Üyesi : Dr. Öğr. Üyesi Ferdi SÖNMEZ	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır.** Aksi halde geçersizdir.

ONAY FORMU

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "İlişkisel ve NoSQL veri tabanı sistemlerinin performans karşılaştırması" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim.(23/06/2018)

Caner SEÇGİN

ÖNSÖZ

Dr. Edgar Frank Codd tarafından 1970 yılında yayınlanan "A Relational Model of Data for Large Shared Data Banks" adlı makale ile hiyerarşik veya gezinme yapısı yerine satır ve sütunlar içeren basit tablolar kullanan ilişkisel veri tabanlarını önerdiğinde mevcut hiyerarşik veya ağ veri tabanlarında işlemler yapmak çok zordu. Ara bir uygulama olmadan verilere erişmek gerektiği için, sorgulama yapan kişi işaretçiler(pointers) gibi karmaşık işlemler ile uğraşmak zorundaydı. Bu temel sorunlar ilişkisel veri tabanlarının ortaya çıkmasına neden oldu. Günümüzde akıllı telefonların artması, sosyal medya kullanımı, metin, görüntü, video, GPS ve sensör verilerinin artması ile veri oluşumu şirketler için petabyte seviyelerine gelmiş ve veri türlerinin bir kısmı artık yapısal olmayan formatlarda oluşmaktadır. Google, Facebook ve Twitter gibi yüz milyonlarca insanın sürekli veri oluşturduğu bu platformların alt yapısında NoSQL veritabanları kullanılmaya başlanmıştır. Bu büyük şirketlerin önderliğinde keşfedilmeye başlanan NoSQL veri tabanları günümüzde birçok farklı veri tabanı modeliyle farklı amaçlarda kullanılmaktadır.
Bu çalışmanın amacı: Yaklaşık 40 yıldır kullanılan ilişkisel veri tabanları ile günümüzde yaygınlaşmaya ve kullanılmaya başlanan NoSQL yani Big data veri depolama sistemlerini performans açısından karşılaştırmak her iki kavramı da artıları ve eksileri ile ele almaktır.
Çalışmada teknik desteği ve yol gösterici tavsiyeleri için Doç.Dr. Metin ZONTUL'a, teknik konulardaki desteği için Alper KURAK'a, yabancı dil çevirilerinde yardımlarını esirgemeyen Recep Duygu ANT'a, her konuda desteğini esirgemeyen anneme, babama ve kardeşime, hayatımdaki varlığı ile gücüme güç katan ve her konuda desteğini esirgemeyen eşim Nur SEÇGİN'e sonsuz teşekkürlerimi sunarım.

Temmuz , 2018 **Caner SEÇGİN**

İÇİNDEKİLER

Sayfa

KISALTMALAR

ACID : Atomicity-Consistency-Isolation-Durability
CAP : Consistency-Availability-Partition Tolerance
CPU : Central Processing Unit
CRM : Customer Relationship Management
DBMS : Database Management Systems
DDL : Data Definition Language
DML : Data Manupulation Language
ERP : Enterprise Resource Planning
JSON : JavaScript Object Notation
SQL : Structured Query Language
RAM : Random Access Memory
RDBMS : Relational Database Management System
VTYS : Veri Tabanı Yönetim Sistemleri

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

İLİŞKİSEL VE NOSQL VERİ TABANI SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

ÖZET

Dr. Edgar Frank Codd tarafından 1970 yılında yayınlanan "A Relational Model of Data for Large Shared Data Banks" adlı makale ile hiyerarşik veya gezinme yapısı yerine satır ve sütunlar içeren basit tablolar kullanan ilişkisel veri tabanlarını önerdiğinde mevcut hiyerarşik veya ağ veri tabanlarında işlemler yapmak çok zordu. Ara bir uygulama olmadan verilere erişmek gerektiği için, sorgulama yapan kişi işaretçiler(pointers) gibi karmaşık işlemler ile uğraşmak zorundaydı. Bu temel sorunlar ilişkisel veri tabanlarının ortaya çıkmasına neden oldu. Günümüzde akıllı telefonların artması, sosyal medya kullanımı, metin, görüntü, video, GPS ve Sensör verilerinin artması ile veri oluşumu şirketler için petabyte seviyelerine gelmiş ve veri türlerinin bir kısmı artık yapısal olmayan formatlarda oluşmaktadır. Google, Facebook ve Twitter gibi yüz milyonlarca insanın sürekli veri oluşturduğu bu platformların alt yapısında NoSQL veritabanları kullanılmaya başlanmıştır. Bu büyük şirketlerin önderliğinde keşfedilmeye başlanan NoSQL veri tabanları günümüzde birçok farklı veri tabanı modeliyle farklı amaçlarda kullanılmaktadır.

Bu çalışmanın amacı: Yaklaşık 40 yıldır kullanılan ilişkisel veri tabanları ile günümüzde yaygınlaşmaya ve kullanılmaya başlanan NoSQL yani Big Data veri depolama sistemlerini performans açısından karşılaştırmak her iki kavramı da artıları ve eksileri ile ele almaktır.

Diğer çalışmalardan farkı ilişkisel VTYS, Doküman tabanlı VTYS, Sütun tabanlı VTYS sistemlerinin aynı hacimdeki veri ile karşılaştırılmasıdır. Sonuç olarak toplu veri yazma, okuma, güncelleme işlemlerinde MongoDB VTYS, silme, gruplama işlemlerinde MySQL VTYS, tek tek yapılan işlemlerde Cassandra VTYS daha performanslı sonuçlar vermiştir. Cassandra VTYS'de toplu işlemlerde tekil alan koşulu zorunlu olduğundan toplu işlemlerde performans yönünden geri kalmıştır. MongoDB dinamik şema oluşturduğundan yazılımcı dostudur.

Anahtar Kelimeler: *NoSQL, İlişkisel Veri Tabanı, MySQL, MongoDB, Cassandra, Veri Tabanı*

THE COMPARISION OF RELATIONAL DATABASE SYSTEM AND NOSQL DATABASE SYSTEM'S PERFORMANCES

ABSTRACT

It was very hard to process data in hierarchical and network databases up to aricle by Dr. Edgar Frank Codd in 1970, "Relational Model of Data for Large Shared Data Banks". Dr. Edgar Frank suggestion was as follows a table system, composed of rows and columns, namely relational database systems.
Some major problems caused to develop relational database systems, also without any secondary applications people can not access to data. Moreover some complicated processes like pointers needed to be used to acess the data
The increase in number of smart phones and the use of social media, text, image, video GPS and also sensor data the production of data reached petabytes and also some of the data produced is not in relational anymore.
Main aim of this study is as follows : The comparison of perfomance of relational databases which is used almost 40 years, and NoSQL databases, namely Big data storage systems, with all advantages and disadvantages.
This study aims to compare all advantages and disadvantages of relational DBMS with Big Data storage systems , namely NoSQL DBMS performances.
Relation based DBMS, Document based DBMS and Column based DBMS performance compared with same size of data, this the main difference of this study from previous ones. As a result MongoDB DBMS perfomed well in read, write, update processes, for software developers dynamical schema creation is more adequate. MySQL DBMS perfomed well in delete and group processes. Cassandra DBMS perfomed well in record base processes which is done one by one, moreover in bulk processes unique field requirement caused poor perfomance.

Key words: *NoSQL, RDMBS, MySQL, MongoDB, Cassandra, Database System*

1. GİRİŞ

Yaklaşık 40 yıldır kullanılan ilişkisel veri tabanı yönetim sistemleri(RDBMS) günümüzde her ihtiyacı karşılamamaktadır. Buradaki temel problemler; teknolojinin hızla gelişmesiyle birlikte artan akıllı telefon kullanımı, sosyal medya kullanımı, sensör, GPS ve log verilerinin hızla artması ve bu teknolojilerin çoğunda oluşan verilerin yapısal ve ilişkisel olmayışı. Örneğin; atılan twitleri ilişkisel bir veri tabanında depolanmaya, işlenmeye çalışılsa ne CPU, ne memory, ne de disk yetmeyecektir. Bunun yanı sıra ilişkisel veri tabanında tutulan bu text formatındaki veriyi nasıl işleyeceğimiz de büyük bir problem oluşturmaktadır. Bu temel problemlerin oluşmasıyla birlikte NoSQL veri tabanları, hem yapısal olmayan verileri tutmak ve işlemek, hem de bu büyük veriyi tutmak ve işlemek için oluşan yüksek tekonoloji maliyetine karşın yatay ölçeklenebilen düşük maliyetli bir teknolojik alt yapı olarak ortaya atılmıştır. Günümüzde veri tabanı yönetim sistemi kullanımında hangi noktada NoSQL veri tabanları daha performanslı ve kullanılmalı, hangi noktada ilişkisel veri tabanları kullanılmalı ve vazgeçilmez olduğu yapılan testler,analizler ile ortaya çıkartılmıştır.

İlişkisel veri tabanları ve NoSQL veri tabanlarının karşılaştırılması ve Big data ile ilgili akademik çalışmalar aşağıdaki gibidir:

MongoDB, MySQL veri tabanına göre insert, update, delete ve select işlemlerinde daha kısa sürelerde işlemleri tamamladı. MongoDB'nin birleştirme(join) işlemlerine izin vermeyen bir veri tabanı olduğu bilinmesine rağmen, bu soruna alternatif vardır. MongoDB ortamında ilişkisel veriler referans dökümanlar(referenced documents) ile sağlanır. Uygulama veri açısından yoğunsa, çok fazla veri depolanmak isteniyorsa ve çok fazla veriler sorgulanmak isteniyorsa MySQL yerine MongoDB seçeriz[1].

İlişkisel veri tabanı yönetim sistemleri, büyüyen veri ihtiyaçlarını, birden fazla bölümleme ve paralelleştirme yeteneklerindeki yetersizliği ile karşılayamaz

durumdadır. NoSQL veri tabanı olan Hadoop, kolayca ölçeklenebilir bir ortamda büyük veri analizi açısından oldukça kabul gören ve kullanılan açık kaynak kodlu bir yazılımdır. Ayrıca Hadoop; güvenilir, düşük maliyetli, dağıtılmış paralel programlamayı destekleyen, yapılandırılmamış veriyi saklama ve analiz edebilme özelliklerine sahiptir[2].

NoSQL veri tabanları birçok farklı türde büyük miktarda veri yönetimi için esnek yollar sunar. İnternet ortamında hızla artan veri yoğunluğu düşünüldüğünde geleceğin teknolojisi ilişkisel veri tabanları yerine NoSQL veri tabanı teknolojisi olabileceği düşünülmektedir. Çeşitli açılardan ilişkisel veri tabanı ve NoSQL veri tabanı yaklaşımı göz önüne alındığında ikisinin de fark yarattığı ve ön plana çıktığı noktalar olduğu görülmektedir. Ancak bazı açılardan ilişkisel bir veri tabanının kullanılmasının çok daha doğru olduğu görülmektedir[3].

MongoDB ve CouchDB karşılaştırılmasında MongoDB, CouchDB'den tek sunucu üzerindeki karşılaştırmalarda ve yatay ölçekleme yapıldığında yani dağıtık mimari üzerindeki karşılaştırmalarda daha performanslı ve daha hızlı olduğunu göstermiştir[4].

Veri okuma için yapılan sorgulama işlemlerinde MongoDB, MySQL veri tabanına göre daha performanslı sonuçlar vermiştir. MySQL veri Silme operasyonlarında MongoDB'e göre daha performanslı. Veri yazma işlemlerinde ise MongoDB, MySQL veri tabanına göre daha performanslı sonuçlar vermiştir[5].

Büyük miktarda depolama ihtiyacı olan yapılar için NoSQL veri tabanları uygundur. Veri yazma işlemlerinde MongoDB, MySQL'e göre daha performanslı sonuçlar vermiştir[6].

Karmaşık olay işleme motorları; Apache Strom, Apache Spark ya da Apache Flink gibi platformlar üzerinde çalışan sistemler, hem ölçeklendirme problemini, hem de büyük veri ve karmaşık olay işleme anlamında daha kapsamlı çözümler üretebilir. Bahsi geçen yeni veri analizi platformlarının kullanıcı ara yüzleri daha basitleştirilirse organizasyonlarda üretim veya hizmet süreçlerinde verimliliği ve hizmet kalitesini arttırıcı çalışmalar ilgili ekiplerce

hızlıca geliştirilip devreye sokulabilir. Bu yaklaşım, sanayi 4.0 vizyonunun gerçek anlamda oluşabilmesi için önemli bir adam olarak görülmektedir[7].

Big data kavramı ve NoSQL veri tabanları, günümüzde çok sık bahsedilen, akademisyenlerin üzerinde birçok araştırma yaptığı ve şirketlerin milyonlarca dolar yatırım yaptığı konulardır. Tezin amacı; NoSQL veri tabanları ile hali hazırda hemen hemen her sektörde, her kurumda kullanılan ilişkisel veri tabanlarını teorik ve pratik olarak inceleyip, henüz bir endüstri standardı haline gelmemiş NoSQL veri tabanlarının hayatımızda nasıl bir rol oynayacağını, hangi sektörlerde ne amaçlarla kullanılacağını tespit etmek ve ilişkisel veri tabanlarının hangi alanlarda kullanılmaya devam edeceğini tespit etmektir.

Bu tez sonunda çıkan sonuçlar ile tez kapsamında bahsedilen konular için kurumların mevcut ihtiyaçlarına en iyi hizmet edecek olan veri tabanı türü, ilişkisel veri tabanları mı yoksa NoSQL veri tabanları mı olduğu sonucu çıkacaktır.

2. VERİ TABANI YÖNETİM SİSTEMLERİ(DBMS)

Veriyi saklama ve işleme ihtiyacı çok eski zamanlara dayanıyor. Henüz bilgisayar çağı başlamamışken firmalar finansal, muhasebesel, satış, satın alma ve Üretim bilgilerini defterlere yazıyorlardı. Müşteri ilişkilerini yönetmek için müşterilerinin bilgilerini ve adreslerini yine defterlerde tutup bayramlarda, yılbaşlarında ve doğum günlerinde tebrik mektupları, kartpostallar gönderiyorlardı. Bugün ise ERP ve CRM sistemleri ve arkasında çalışan ilişkisel veri tabanı yönetim sistemleriyle birlikte bu operasyonların tamamı dijital ortamda yapılmaktadır. Bu dijital çözümler sayesinde finansal durumlar ve müşteri ilişkileri hem anlık veriler sorgulanıp anlamlandırılarak çeşitli aksiyonlar alınıyor, hem de geçmiş veriler ile ilişkilendirilip gerekli aksiyonlar alınabiliyor. VTYS kavramı ilk olarak 1960 yılında tanımlandı. Yapısal olarak bütün veri tabanları aynı değildir. Temel görevi, istenilen bilgileri saklamak olmasın rağmen yapısal olarak farlılıklar içermektedir. Veri modeli, verilerin depolanması, işlenmesi, veriler arası ilişkilerin kurulma biçimine göre VTYS farklı gruplara ayrılır. Bunlar; Hiyerarşik Veri tabanları, Ağ Veri tabanları, İlişkisel Veri tabanları, NoSQL Veri tabanlarıdır[8].

2.1 CAP Teoremi

Veri tabanı teknolojilerinin gelişmesiyle birlikte veri tabanlarının bir kısmı dağıtık mimarilerden oluşur. Dağıtık mimarilerde bilinmesi gereken en önemli husus CAP Teoremidir. CAP teoremi, 1998 yılında California Üniversitesi bilgisayar bilimcisi Eric Brewer tarafından ortaya atılmıştır. CAP teoremi Consistency(tutarlılık), Availability(erişebilirlik), Partition Tolerance(bölümleme) kelimelerinden oluşur. Brewer'ın teorisine göre, bu üç özellik aynı anda bir veri tabanında bulunamaz. Günümüzde bu durum güncelliğini korumaktadır[9].

i. Consistency

Her kullanıcı, her istemci her zaman aynı veriyi görür. Tüm kullanıcılar her zaman en güncel veriyi görür. Bu sayede tutarlılık sağlanmış olur.

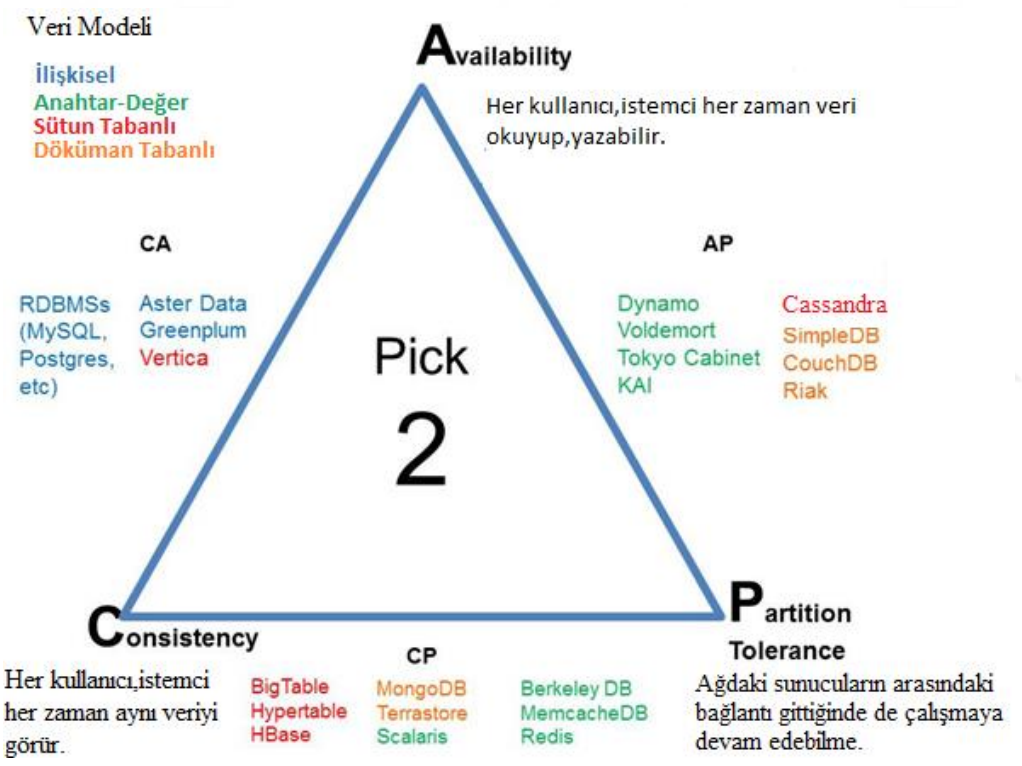
ii. Availability

Her kullanıcı ve istemci her zaman veri okuyup yazabilir.

iii. Partition Tolerance

Ağdaki sunucuların arasındaki bağlantı gittiğinde de sistem çalışmaya devam eder[10].

Şekil 1'de CAP teoremi bir üçgen ile gösterilmiştir.



Şekil 2.1: CAP Teoremi gösterimi

2.2 Hiyerarşik Veri Tabanları

Bu veri tabanı modelini 1965 yılında IBM ve Kuzey Amerika Havacılık şirketi birlikte geliştirmiştir. Tarihte ilk kullanılan veri tabanı modelidir. Bu modelde veriler ağaç(tree) yapısında tutulmaktadır. Kök olarak bir kayıt ve bu kayıta bağlı alt kayıtlar oluşur. Varlıklar arasındaki ilişkileri kurarken her varlığın tek bir varlığa bağlanması gerekir. Aynı varlık birden fazla ilişkide kullanılacaksa

tekrar oluşturmak gerektiği için gereksiz veri tekrarına neden olur. Tezin amacı, bu veri tabanlarını incelemek olmadığından üzerinde daha detaylı durulmamıştır[11].

2.3 Ağ Veri Tabanları

Hiyerarşik veri tabanı modelinin yetersiz kalmasından dolayı, General Electric şirketinde çalışan Charles Bachman tarafından verileri birbirine bağlayarak çalışan bu model geliştirilmiştir. Ortak bilgi alanları tanımlanarak birden fazla veri ile ilişki kurulabilir. Ebeveyn-çocuk(parent-child) ilişkisinden farklı olarak her kaydın birçok ebeveyn ve birçok çocuk kaydı bulunabilir. En karmaşık veri tabanı modelidir. 1970'li yıllarda ve 1980'li yılların ilk yarısında kullanılmıştır. Tezin amacı, bu veri tabanlarını incelemek olmadığından üzerinde daha detaylı durulmamıştır [11].

2.4 İlişkisel Veri Tabanı Yönetim Sistemleri(Rdbms)

İlişkisel veri tabanı teorisi, 1970 yılında IBM şirketinde çalışan Dr. Edgar F. Codd tarafından önerilmiştir. İlişkisel veri tabanının temel mantığı, büyük hacimlerdeki verilerin güvenli bir şekilde tutulduğu, verilerin tutarlı bir bütünlük içinde saklandığı, gerektiğinde hızlı bir şekilde doğru veriye ulaşılabildiği bir ortam olmasıdır. Bu doğrultuda ilk ilişkisel veri tabanı, 1979'da Oracle şirketi tarafından geliştirilen Oracle 2 veri tabanıdır. Bu sistemler kendi içerisinde verilerin birbirleri ile ilişki kurmasıyla daha tutarlı ve daha güvenli sorgulanmasına ve kullanılmasına olanak sağlamaktadır. İlişkisel veri tabanı sistemlerinin yaygın olarak kullanılmasının temel sebepleri bunlardır[12].

2.4.1 Acid

RDBMS'ler işlem(transaction) tabanlı çalışan sistemlerdir. Bu işlemlerin stabil çalışması ve veri bütünlüğü için ACID(Atomicity, Consistency, Isolation, Durability) kuralları bulunur.

2.4.1.1 Atomicity(Bölünmezlik)

İşlemler(Transaction) bir bütün olarak görülür. İşlem sırasında birden fazla tablodaki verinin güncellenmesi gerçekleşiyor ise tüm bunların hepsi birden başarılı olacaktır veya başarısız olacaktır.

2.4.1.2 Consistency(Tutarlılık)

İşlem(Transaction) sonucunda veri tabanındaki verinin geçerli durumunun, bir sonraki geçerli duruma geçmesidir. Yani işlem tam anlamıyla bitene kadar işlemden etkilenecek verilerin tamamı işlemden bir önceki geçerli değerleri gösterecektir.

2.4.1.3 Isolation(Ayrı tutma)

Aynı anda, aynı veri üzerinde birden fazla transaction değiştirme gereksinimi olabilir. Transaction'ların birbirlerinin işlemlerinden etkilenmemesi için işlemler seri olarak yapılması gerekir. Transaction başarılı veya başarısız olarak sonuç dönünceye kadar etkilenecek veri setleri kilitlenir.

2.4.1.4 Durability(Dayanıklılık)

Transaction sırasında oluşacak bir hataya karşı, sistemin kendisini bir önceki geçerli duruma döndürebilme kabiliyetidir[url-1].

2.4.2 İlişkisel veri tabanlarının durumu

Veri tabanı yazılımları son 40 yıldır işletim sistemi yazılımları ile birlikte yazılım teknolojileri pazarında en büyük paya sahiptir. Dijitalleşme ile dijital veri üretimindeki artış, veri yönetim teknolojilerinin bu alandaki liderliklerinin süreceğini gösteriyor. Günümüzde en çok kullanılan veri tabanı yönetim sistemleri(VTYS) ilişkisel veri tabanlarıdır. İlişkisel veri tabanlarını yazılımlarının bir çoğu kapalı kaynak kodludur. Kapalı kaynak kodlu veri tabanı yazılımları, geliştirilen şirket tarafından belirli periyotlarla güncellenmektedir. Bu veri tabanları; Oracle, Microsoft SQL Server, IBM DB2, Microsoft Access, Teradata veri tabanlarıdır. İlişkisel veri tabanlarının bir kısmı ise açık kaynak kodlu yazılımlardır. Bunlar; MySQL, PostgreSQL. Bu veri tabanları Global ortak akıl ve paylaşım ile yazılım ve bilgisayar mühendisleri tarafından geliştirilmektedir[url-2][url-3].

2.4.3 Sql

İlişkisel veri tabanlarındaki veriler sorgulanarak, verilerden anlamlı bilgiler çıkarılmaya çalışılmaktadır. İlişkisel veri tabanlarında sorgulama yapabilmek ve istenen veri setini oluşturabilmek için SQL(Structured Query Language) sorgulama dili kullanılmaktadır. SQL, 1970 yılında IBM şirketinde çalışan Dr. Edgae F. Codd tarafından ilişkisel veri tabanı modeli üzerine kurularak geliştirilmiş bir programlama dilidir.

Tüm ilişkisel veri tabanlarında kullanılan standart bir sorgulama komut kütüphanesi ANSI(American national standarts instutite) tarafından geliştirilmiştir. ANSI 1980 yılında oluşturulmuştur[12].

2.4.3.1 Veri işleme dili(Data manuplation language, DML)

Bir veri tabanı üzerinde kayıt ekleme(insert), kayıt güncelleme(update), kayıt silme(delete), veya kayıt birleştirme(merge), işlemleri için kullandığımız komutlardan oluşur. Bu tip komutlar, veri tabanına commit veya rollback komutları göndermeden tablolar üzerinde kalıcı şekilde işlem gerçekleştirmezler.

2.4.3.2 Veri tanımlama dili(Data definition language, DDL)

Dml işlemleri ile veri tabanı üzerindeki objeler tanımlanır. Bu objeler ile tablo, user, schema, view, index gibi objeler oluşturulabilir(create), silinebilir(drop) veya değiştirilebilir(alter).

2.4.3.3 Veri denetleme dili(Data control language, DCL)

Kullanıcılara veya rollere nesneler üzerinde yetki verilmesini(grant) veya yetkinin geri alınmasını(revoke) sağlar.

2.4.3.4 İşlem denetleme dili(Transaction control language, TCL)

Bir veri tabanı üzerinde işlemleri kalıcı hale getirmek için kullanılan commit ya da yapılan işlemin veri tabanında kalıcı olmadan geri alınması için kullanılan rollback işlemleridir.

2.4.4 İlişkisel veri tabanı nesneleri

İlişkisel veri tabanı yönetim sistemlerinde(İVTYS) veriler tablolarda(Table) tutulur. Tablolar içerisindeki alanlar(column, field) ile veriler ayrıştırılır.

Örneğin; bir müşteri tablosu düşünelim; bir alanında isim soy isim, bir alanında doğum tarihi, bir alanında tc kimlik numarası olabilir. İlişkisel veri tabanlarında tablolar arasında ilişkiler kurulur. İVTYS mantığına göre müşteri tablosunda o müşteriye ait alışverişler yer almaz. Bunun yerine Alışveriş adında bir tablo olur. Alışveriş tablosunda ise ürünü alan müşteri, satış tutarı, kdv tutarı, alışveriş tarihi gibi bilgiler ayrı alanlarda tutulur. Örnek Alışveriş tablosu Çizelge 2.1'de gösterilmiştir.

Çizelge 2.1: Örnek Alışveriş Tablosu

ALISVERIS				
ALISVERIS_ID	**SATIS_TUTARI**	**KDV_TUTARI**	**ALISVERIS_TARIHI**	**MUSTERI_ID**
1	10.8	0.80	10.05.2018	1
2	30.30	2.24	28.04.2018	2
3	66.50	10.14	19.04.2018	3
4	21.00	0.21	15.04.2018	1
5	37.50	2.78	19.04.2018	4

Çizelge 2.2: Örnek Müşteri Tablosu

MUSTERI			
MUSTERI_ID	**ISIM_SOYISIM**	**DOGUM_TARIHI**	**TC_KIMLIK_NO**
1	CANER SEÇGİN	01.11.1985	19742552719
2	METİN ZONTUL	16.03.1966	59382552876
3	ALPER KURAK	17.07.1935	33768553219
4	TAMER KARAGÖZ	25.04.1978	24242334819

Örnek müşteri tablosu yukarıda Çizelge 2.2'de gösterilmiştir. Müşteri ve alışveriş tablosu arasındaki ilişki, 1'den çoğadır(1-n). Yani 1 müşteri birden çok alışveriş yapabilir, ama 1 alışveriş birden çok müşteri tarafından yapılamaz. Bu ilişki, İVTYS'de yabancı anahtar(foreign key,fk) ile tanımlanır. Yabancı Anahtar(Fk) Şekil 2.1'de gösterilmiştir.

Şekil 2.2: Müşteri ve Alışveriş tablosu arasındaki yabancı anahtar(fk)

Yukarıda bahsedilen basitçe ilişkisi gösterilen iki tablo yabancı anahtar ile bağlanmıştır. İlişkisel veri tabanlarında tablolara çeşitli kısıtlamalar(constraint) eklenir. Bu kısıtlamalar sayesinde bazı alanların benzersiz(unique) olması, bazı alanların mutlaka dolu(not null) olması sağlanır. Yukarıda ilişkisi gösterilen Müşteri tablosunda MUSTERI_ID, alışveriş tablosunda ALISVERIS_ID alanları birincil anahtardır(primary key). Birincil anahtar tanımlaması o alanın, hem benzersiz, hem de mutlaka dolu olmasını sağlar.

İlişkisel veri tabanları çeşitli ihtiyaçları sağlamak için kendi içerisinde ikiye ayrılır. Bunlar; gerçek zamanlı operasyonel verilerin işlenip tutulduğu veri tabanları(Online Transactional Processing) ve verilerin analitik işlemler yapmak için tutulup işlendiği veri ambarları, OLAP(Online Analytical Processing) veri tabanlarıdır.

2.4.5 Oltp vtys(Online transactional processing)

OLTP veri tabanları, şirketlerin, kurumların mevcut operasyonları yürütürken, anlık olarak oluşan verilerin depolandığı veri tabanlarıdır. Bu veri tabanlarında, anlık olarak veri işleme(Data manipulation language, DML) işlemleri yapılır. Bu işlemler; veri yazma(insert), güncelleme(update), silme(delete) işlemleri yapılır. Örnek vermek gerekirse; bir bankacılık sisteminde, bankaya gidip yeni hesap açtırdığınızda hesapların ve kişilerin tutulduğu tablolara bilgileriniz eklenir(insert). Bu banka hesabından para çektiğinizde veri tabanında bir update işlemi yapılır ve hesabınıza tanımlı tutar güncellenir. Bu bankadaki bazı bilgilerinizin kişisel verilerin korunması kanunu kapsamında silinmesi gerekiyorsa delete işlemi yapılır. Gün içerisinde yapılan işlemlerin tamamı bu veri tabanlarında DML işlemleri yapılarak kayıt altına alınır ve kullandığımız atmlerden veya vezne ekranlarından hesaplarımız ile ilgili güncel durumları bu veri tabanından anlık olarak görürüz. Bu denli büyük ve kritik sistemlerde yapılan bu değişikliklerin tamamı ayrıca bir log tablosunda tutulur. Bu log tablolarına genelde update veya delete işlemi yapılmaz. Yapılan işlemler ile ilgili detay bilgi bu tablolara insert edilir. Örneğin; hesap hareketleri_log gibi bir tablo ismi ile oluşturulan bir tabloda hangi saatte, hangi anda, hangi hesapta, kim tarafından update, delete, insert bilgisi olduğu detaylı bir şekilde tutulur. Örnek log tablosu Çizelge 2.3'de gösterilmiştir.

Çizelge 2.3: Hesap Hareketleri ile ilgili log tablosu

LOG_ID	HESAP_ID	ISLEM_TARIHI	ISLEM_TURU	ISLEM_YAPAN	ILK_DURUM	SON_DURUM
1	1544330300	12.03.2018 11:04:12	UPDATE	CANER	1000	950
2	1544330301	12.03.2018 11:04:13	INSERT	AHMET	0	5000
3	1544330302	12.03.2018 11:04:13	UPDATE	MEHMET	2000	2500
4	1544330303	12.03.2018 11:04:15	UPDATE	METİN	850	820
5	1544330304	12.03.2018 11:04:17	INSERT	FATİH	0	300
6	1544330300	12.03.2018 11:04:14	DELETE	CANER	950	

OLTP sistemler çevremizdeki hemen hemen tüm işletmelerde mevcuttur. Kişilerin yaptığı bankacılık işlemleri, mağazalarda yaptığımız alışveriş işlemleri, otobüs, metro gibi toplu taşıma işlemleri, taşımacılık ve rezervasyon işlemleri, stok kontrol işlemleri, kişilerin demografik bilgilerinin işlemleri vb.

OLTP veri tabanlarında sorgulama işlemleri sisteme yük getirmemesi için çok fazla yapılmaz. Eğer anlık bir veriye(data) bakılmak isteniyorsa OLTP sisteminin anlık kopyasının tutulduğu(replika, mirror) veri tabanından sorgulanır. Eğer geçmişe yönelik bir sorgulama, analatik bir sorgulama yapılacaksa bu sorgulamalar veri ambarı(DWH) ortamlarından veya Olap ortamlarından çekilir. Buradaki asıl amaç anlık işlemler(transaction) ile yoğun çalışan bir sisteme bir de sorgulama yükü getirerek yormamaktadır.

2.4.6 Olap(online analytical processing) ve veri ambarları(data warehouse)

OLAP veri tabanları ve veri ambarları şirketlerin, kurumların mevcutta yürüttüğü operasyonları çeşitli konular özelinde mevcut veri setlerini analatik olarak incelenmesine yarayan sistemlerdir. İlk olarak üst yönetimin alacağı kararlar için karar destek sistemleri olarak ortaya çıkmıştır. Günümüzde çeşitli departmanların kritik kararlar alması ve mevcut işleyişi kümüle olarak analiz edilip incelenmesi için kullanılmaktadır. Genelde geçmişe yönelik analizler yapmak için kullanılır. Örneğin; son 6 aylık satış trendi, il, ilçe veya bölgeye göre gruplanmış bir şekilde gösterimi veya en karlı ürünlerim neler? Hangi mağaza veya şubelerimde hangi saatlerde yoğunluk yaşanmakta? Gibi soruları hızlı bir şekilde bu sistemlerden cevaplanabilir.

Veri ambarlarının temel amacı; OLTP sistemlerine yük getirmemek ve farklı Oltp sistemlerinde bir konu özelinde duran veri setlerini bir araya getirmektir. Örneğin; müşteri ilişkileri yönetim sisteminin veri tabanı ile faturalama sisteminin veri tabanı dataları çeşitli veri transfer yöntemleri(ETL) ile veri

ambarı ortamlarına belirli periyotlarla taşınır ve bu veri setleri birleştirilerek veriler anlamlandırılır. Bir müşterinin demografik bilgileri ile yapılan satışın ilişkisi bu sistem sayesinde ortaya çıkmış olur.

OLTP sistemlerden Veri Ambarlarına veri transferi işlemleri genellikle gece yapılır. Bunun sebebi gün içerisinde aktif olarak çalışan OLTP sistemlerine bir de veri ambarlarına veri atmak için gelecek sorgu maliyetlerini engellemektir. OLTP sistemlerinde genellikle geceleri yoğunluğu az olacağı için bu süreçler kolaylıkla halledilmektedir. Veri ambarlarına, Oltp sistemlerden gece veriler taşınır ve bu veriler üzerinde çeşitli gruplama, toplama, çıkarma, pivot işlemleri yapılarak veri tabanı tabloları oluşturulur. Gün içerisinde iş birimleri veya karar mercileri çeşitli raporlama ve veri analizi uygulamaları ile gece oluşturulmuş bu veri setlerini sorgularlar. OLTP ve OLAP karşılaştırılması Çizelge 2.4'de gösterilmiştir.

Çizelge 2.4: OLTP, OLAP karşılaştırması

OLTP	**OLAP**
Veri değişimi işlemleri yoğunluktadır.(DML)	Gün içerisinde genelde sorgulama yapılır.
Anlık yapılan işlemler ile veri setleri(tablolar) oluşur.	Belirli zaman aralıklarıyla(genelde her gece) veri setleri(tablolar) oluşturulur.
Güncel veriler saklandığı için veri miktarı az.	Geçmişe yönelik veriler saklandığı için veri miktarı çoktur.
Uygulama odaklı çalışır.	Konu odaklı çalışır
Günlük operasyonlar için kullanılır.	Karar destek sistemidir.
Kısa ve basit işlemler yapılır.	Karmaşık sorgulamalar yapılır.
Verinin küçük bir kısmı ile ilgili	Daha büyük veriye erişim

2.5 NoSQL Veri Tabanları

Teknolojinin gelişmesiyle birlikte, akıllı telefonların ve uygulamaların artması çok çeşitli, çok hızlı artan ve çok büyük veri türlerini hayatımıza kattı. Günümüzde cihazlarda türeyen veriler xml, json, metin, ses, görüntü, web sayfası, pdf, video, fotoğraflar gibi verileri işlemek ve bilgiye dönüştürmek için alışılagelmiş ilişkisel veri tabanları ve yapısal veri tipleri yetersiz kalıyor ve çok çeşitli sorunlara sebep oluyordu. Bu gelişmelerle birlikte NoSQL veri tabanları son yıllarda bilişim dünyasında hızla yaygınlaştı ve ilişkisel veri tabanı yönetim sistemlerinden farklı ve alternatif bir yaklaşım olarak ortaya çıktı. Bu

yaygınlaşmaya çok büyük teknoloji şirketleri önderlik etmiştir. Google devasa boyuttaki verisini Big Table adını verdiği NoSQL veritabanı sistemi ile Amazon ise DynamoDb ismini verdiği NoSQL veri tabanı ile verilerini işlemektedir[13].

NoSQL veri tabanların İVYTS'den farklı olarak yatay ölçeklenerek mevcut sistem kaynaklarına eklemeler yaparak verileri böler, kopyalarını dağıtık sistemin farklı parçalarına ekler. Böylelikle bir kayıp yaşansa da verinin tamamı kaybedilmez ve sisteme gelecek yük dağıtılmış olur.

NoSQL sistemlerde İVTYS'de olduğu gibi ACID kurallarını garantilemez. ACID yerine BASE(basically, available, soft state, eventually consistent) kurallarını garantiler. Basically availablitiy ile sistemin sürekli açık ve erişilebilir olması, soft state, yani veriler her zaman güncel olmayabilir; çünkü başka bir kopyada ilgili veriler henüz güncellenmemiş olabilir. Eventually consistent, yani gün sonunda veya belirli periyotlarla sunucuların birbirleriyle iletişim kurarak birbirlerini eşitlemesi, haberdar etmesi[4].

NoSQL veri tabanları kendi içerisinde 4 ana tipe ayrılır. Bu tipler arasında veri tutarlılığı ve erişimi ile ilgili farklı özellikleri vardır.

2.5.1 Anahtar-değer tabanlı(Key value store)

Veri, bir anahtar ve buna karşılık gelen değer şeklinde kayıt edilir. Sorgulama yaparken bu anahtar aracılığı ile bilgiye ulaşılır. Kolon kavramı yoktur. anahtar-değer tabanlı veri tabanları arasında en yaygın kullanılan Redis'dir. anahtar-değer tabanlı VTYS sistemleri; Redis, Amazon DynamoDB, Memcached, Hazelcast, Riak KV, OrientDB, Oracle NoSQL, Oracle Berkeley DB.

2.5.2 Sütun tabanlı(column store)

Bu tür veri tabanı teknolojileri genelde çok büyük miktarda veri tutma amacı ile kullanılır. Anahtar-değer veri tabanlarında olduğu gibi bir anahtar bulunur; fakat bu anahtar sadece ilgili alanı(field) adreslemez, o sütunun ilişkili olduğu sütun grubunu adresler. En yaygın kullanılan; sütun tabanlı VTYS Cassandra'dır. Sütun tabanlı VTYS sistemleri; Cassandra, HBase, Hypertable, Amazon simpledb.

2.5.3 Doküman tabanlı(document store)

Anahtar-değer veri tabanlarına benzer bir diğer yaklaşım olan döküman tabanlı veri tabanlarında, değer kısmı da veriyi anahtar-değer olarak tutar. Bu iç içe geçmiş yapı sayesinde birbiriyle alakalı bütün veriler aynı yerde tutulur ve ulaşılması çok kolaydır.

Bu sistemlerde kayıtlar doküman olarak tutulur. Genel olarak Json formatında tutulur. Bu tip VTYS'leri arasında en çok kullanılanı MongoDB'dir. Doküman tabanlı VTYS sistemler; MongoDB, Amazon DynamoDB(Çoklu model olduğundan hem Anahtar-değer hem de Doküman tabanlı kullanılabilir.), CouchBase, CouchDb, Microsoft Azure Cosmos DB, Marklogic, Firebase Realtime Database, RethinkDB, Cloudant[url-3].

2.5.4 Çizge tabanlı(graph dbms)

Bu veri tabanı türünde veriler çizge şeklinde tutulur. Diğer veri tabanlarından oldukça farklı bu türde çizge yapısındaki verileri performanslı bir şekilde sorgulamamıza yarar. En sık kullanılanı Neo4j VTYS'dir. Örnek sistemler; Neo4J, Microsoft Azure Cosmos DB(çoklu model olduğundan doküman tabanlı hemde çizge tabanlı kullanılabilir.) Datastax Enterprise, Girapj, JanusGraph, InfiniteGraph.

2.6 Veri Tabanı Yönetim Sistemlerinin Kullanım Durumu

Veri tabanı yönetim sistemleri(DBMS) hakkında bilgi toplamak ve sunmak için kurulan DB-Engines verilerine göre en çok kullanılan veri tabanları Şekil 7'de gösterilmiştir. Bu platform bilgileri aylık olarak güncellediğinden Haziran 2018 itibariyle liste gözükmektedir. Buna göre en çok kullanılan VTYS'ler hala ilişkisel veri tabanlarıdır. Fakat liste incelendiğinde NoSQL veri tabanlarının hızlı bir şekilde yükseldiğini ve kullanım alanlarının arttığı gözlemlenmektedir. Bu çalışma kapsamında MySQL, MongoDB ve Cassandra kullanılmasının amacı hem en çok kullanılan VTYS'ler hem de açık kaynak kodlu olmalarıdır. 2018 Haziran ayı verilerine göre en sık kullanılan VTYS'ler Şekil 2.2'de gösterilmiştir.

343 systems in ranking, June 2018

Rank			DBMS	Database Model	Score		
Jun 2018	May 2018	Jun 2017			Jun 2018	May 2018	Jun 2017
1.	1.	1.	Oracle	Relational DBMS	1311.25	+20.84	-40.51
2.	2.	2.	MySQL	Relational DBMS	1233.69	+10.35	-111.62
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1087.73	+1.89	-111.23
4.	4.	4.	PostgreSQL	Relational DBMS	410.67	+9.77	+42.13
5.	5.	5.	MongoDB	Document store	343.79	+1.67	+8.79
6.	6.	6.	DB2	Relational DBMS	185.64	+0.03	-1.86
7.	7.	↑ 9.	Redis	Key-value store	136.30	+0.95	+17.42
8.	↑ 9.	↑ 11.	Elasticsearch	Search engine	131.04	+0.60	+19.48
9.	↓ 8.	↓ 7.	Microsoft Access	Relational DBMS	130.99	-2.12	+4.44
10.	10.	↓ 8.	Cassandra	Wide column store	119.21	+1.38	-4.91

Şekil 2.3: VTYS kullanım sıralaması

3. MYSQL VERİ TABANI

MySQL VTYS 1995'de ilk yayınlandığından günümüze hızlı bir şekilde yaygınlaşmıştır. Özellikle, php ile kodlanan web sitelerinde sıkça kullanılmıştır. MySQL VTYS günümüzde en çok kullanılan açık kaynak kodlu ilişkisel veri tabanıdır. MySQL veri tabanında Sql dili kullanılır. MySQL, ACID uyumlu bir veri tabanıdır. MySQL arka tarafta veriyi fiziksel veri dosyalarında tutar. Kullanıcılar veya veri tabanı kullanıcıları bu fiziksel veri dosyaları ile ilişkilendirilmiş tablespaceler içerisinde oluşturulmuş tablolar üzerinde duran veriler ile işlemler yapar. MySQL veri tabanı hem OLTP yapılar, hem de OLAP yapılar için kullanılmaktadır. OLTP yapılarda genelde tablolar üzerine indeksler koyulur. İndeksler gelen bir veya birkaç işlemi daha hızlı yapmak veya bir kayıt sorgulama gibi işlemleri daha hızlı yapmak için oluşturulur. Örneğin; bir market sistemindeki alışveriş tablosuna her bir kasadan 1'er 1'er alışveriş işlemleri veri tabanına işlenmektedir. OLAP yapılarda ise yine MySQL veri tabanının desteklediği bölümleme(partition) yapısı kullanılır. Bu partition yapısı sayesinde veriler aynı tabloda olmasına rağmen belirli aralık veya listeler kullanılarak disk üzerinde ayrı yerlere yazılır. Bu sayede ilgili bölüm hızlıca kullanıcıya sunulur. Örneğin; bir alışveriş tablosunun veri ambarındaki yapısında genelde 1 aylık veya geçmiş 3, 6 aylık toplam alışveriş veya KDV tutarları sorgulanacağı için alışveriş tablosundaki alışveriş tarihine Range(aralık) partitionu tanımlanır. Bu sayede ayrı bir alanda duran 1 aylık alışveriş verisi tüm tablo içerisinde aranıp, bulunmakla uğraşılmaz direkt ilgili partitiondan getirilir. MySQL, Facebook, Google, Twitter, Uber ve Booking.com şirketlerde kullanılmaktadır.

3.1 MySQL Kurulum

MySQL VTYS kurulumu Ubuntu 16.04 versiyonlu işletim sistemine aşağıdaki şekilde kurulmuştur:

```
csecgin@csecgin-VirtualBox:~$ sudo apt-get update
```

csecgin@csecgin-VirtualBox:~$ sudo apt-get install MySQL-server

csecgin@csecgin-VirtualBox:~$ MySQL_secure_installation

Kurulum yukarıdaki adımlar ile tanımlandı. MySQL'in doğru bir şekilde çalışıyor olduğunu aşağıdaki kod ile test ediyoruz.

csecgin@csecgin-VirtualBox:~$ MySQLadmin -p -u root version

Enter password:

MySQLadmin Ver 8.42 Distrib 5.7.22, for Linux on x86_64

Copyright(c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Server version 5.7.22-0ubuntu0.16.04.1

Protocol version 10

Connection Localhost via UNIX socket

UNIX socket /var/run/MySQLd/MySQLd.sock

Uptime: 59 min 26 sec

Threads: 1 Questions: 2 Slow queries: 0 Opens: 107 Flush tables: 1 Open tables: 26 Queries per second avg: 0.000

Ubuntu 16.04 versiyonlu işletim sistemine yukarıdaki gibi MySQL veri tabanı kurulmuş, tez için yapılacak testler için kullanılmıştır.

3.2 MySQL Veri Tipleri(Data Type)

MySQL veri tabanında veriler tablolarda satır satır tutulur. Bu satırlar içerisindeki bilgiler columnlar(sütun) ile ayrılır. Bilgilerin içeriğine göre her column'a bir veri tip tablo oluşturulurken tanımlanır. MySQL veri tabanındaki veri tipleri Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1: MySQL Veri Tipleri

Veri Tipi	Açıklama
char(n)	Belirtilen n(0-255) sayısı kadar alfa nümerik ifadeler için kullanılır. n'de belirtilen karakter sayısı kadar diskte yer kaplar.
date	1000-01-01 yılından 9999-12-31 yılına kadar değişebilen tarih bilgisi tutar.
mediumblob mediumtext	En fazla 16777215 karakter uzunluğunda metin veya binary veri tutar.
nchar(n)	UNICODE karakterler içerebilir. 2000 byte sabit uzunlukta veri tutar.
numeric(p,s) decimal(p,s)	Tamsayı ve virgüllü sayılar için kullanılan sayısal veri tipidir. Tam kısım(p) ve ondalık kısmın(s) ile belirlenir. Örneğin;(2,1) olarak tanımlanan bir sütunun tam sayı kısmı 2 ondalık kısmı 1 basamaklı olacaktır(10.3 gibi).
nvarchar	En fazla 255 karakter ve değişken uzunlukta karakterler tutar UNICODE içerebilir.
text	En fazla 65535 karakter alabilen metin verisi tutar
varchar(n)	Belirtilen n(0-255) sayısı kadar alfa nümerik ifadeler için kullanılır. N'de belirtilen karakter sayısı kadar değil, ilgili kaydın karakter uzunluğu kadar diskte yer kaplar.
smallint	-32768 ile 32767 aralığında işaretli ve işaretsiz tamsayıları tutar.
timestamp	01.01.1970 00:00:00 ile 12.31.2037 23:59:59 arasında zaman bilgisi tutar. Bu bilgi belirtildiği gibi gün, ay, yıl, saat, dakika ve saniye şeklinde tutulur.
tinyblob, tinytext	255 karakter veya daha az uzunlukta metin veya binary veri tutar.
tinyint	-128 ile 127 aralığında işaretli ve işaretsiz tamsayıları tutar

3.3 MySQL Nesneleri

MySQL veri tabanında veriler satır ve sütun olarak tablolarda tutulur. Bir veya birden fazla tablodan veriler tek bir tablodaymış gibi çekmek istenirse view kullanılır. View içerisinde bir veya birden fazla tablonun datasının oluştuğu sql bulunur. Tüm bu SQL'i çalıştırmak yerine tanımlanmış view çalıştırılarak ilgili veri seti sorgulanır. Sorguların performaslı çalışması için tablolara indexler eklenir. İndex genellikle number(sayı) alanına tanımlanır. İndex kitaplardaki veya tezlerdeki içindekiler kısmı gibidir. Hangi verinin nerede olduğu bilgisini tutar. Bu sayede verilere çok hızlı bir şekilde erişim sağlanır. Synonyms ile veri tabanı nesnelerine alternatif isimler verilir. Sequence, Birincil anahtar(primary key) oluşturulmasını ve belirli bir sırayla ilgili kolondaki verinin artmasını sağlar. Örneğin; her kayıt geldiğinde id alanı 1'er 1'er artar.

MySQL VTYS'de tablo oluşturma;

```
CREATE TABLE test.calisanlar(
id number,
isim varchar,
soyisim varchar,
dogum_tarihi date,
);
```

MySQL Vtys'de index oluşturma;

```
CREATE INDEX idx_calisan
ON test.calisanlar(id);
```

4. CASSANDRA

Cassandra ilk kez 2008 yılında Apache Software Foundation tarafından Google BigTable ve Amazon DynamoDb'den esinlenerek yayınlanmıştır. Apache Cassandra açık kaynak kodlu sütun tabanlı bir VTYS'dir. Java programlama dili ile yazılmıştır. Cassandra'da verileri sorgulamak ve işlemek için SQL diline çok benzeyen CQL(Cassandra query language) kullanılır. Windows, Linux, OS X, BSD işletim sistemlerinde çalışabilir. İlişkisel veri tabanlarından farklı olarak şema(schema) olmaksızın veriler JSON yada XML formatında tutulur. C#, C++, Go, Java, JavaScript-Node.js, PHP, Ruby, Scala yazılım dillerini destekler. Sharding methodu ile verileri bölümleme(partitioning) yapar.

İlişkisel veri tabanlarından farklı olarak yabancıl anahtar(foreign keys) kullanılmaz. In-Memory kapasitesi yoktur. Toplu(Bulk) işlemleri desteklemez. Eğer bir veriyi silmek veya güncellemek istiyorsanız o veriyi belirten koşulu tabloya tanımlanmış birincil anahtar(primary key) kolonu ile belirtmek zorundasınız. Cassandra'da İVTYS'de olduğu gibi 'like' ile veri aramak mümkün değildir. Yine İVYTS'de olduğu gibi tabloları join işlemleri ile birleştirmek mümkün değildir.

Cassandra CAP(consistency, availability, partition tolerance) teoreminden Availability ve Partition Tolerance'ı sağlar. Yani hem her istemcinin(client) her zaman veri yazma, okuma ve güncellemesine izin verir, hem de veriler başka sunuculara dağıtılarak yatak ölçekleme yapılır ve sisteme gelecek yükü dağıtır(yatay ölçekleme). Ana düğüm(master node) konsepti Cassandra'da yoktur. Bu sayede Master-Slave mimarisinde Master düğümde bir problem olduğunda çıkabilecek sorunların önüne geçilir. Tüm düğümler(node) birbirleriyle iletişim halindedir. Verileri yerleştirmek için dağıtık hash tabloları kullanır. Cassandra'da veriler replica(kopyalama) desteği sunar ve replica işlemleri düğümler arasında otomatik olarak gerçekleştirilir. Veri tabanının oluşturulması esnasında kopya sayısı tanımlanır[url-3][14][15].

4.1 Cassandra Kurulum

Cassandra, Java ile geliştirildiği için çalışırken JVM'e ihtiyaç duyar. Bu sebeple Cassandra kurulumlarını yapmadan önce Ubuntu işletim sisteminde Java kurulmuştur.

Ubuntu 16.04 VPS işletim sistemi olan makinamıza Java 8 yüklüyoruz;

csecgin@csecgin-VirtualBox:~$ sudo apt-get install default-jdk

Java 8'in kurulu olduğunu doğrulamak için aşağıdaki gibi kontrol ediyoruz:

csecgin@csecgin-VirtualBox:~$ java –versionopenjdk version "1.8.0_171"

OpenJDK Runtime Environment(build 1.8.0_171-8u171-b11-0ubuntu0.16.04.1-b11)

OpenJDK 64-Bit Server VM(build 25.171-b11, mixed mode)

Java versiyonun sorunsuz bir şekilde kurulduğu görüldükten sonra Cassandra Repository(depo) kurulumu adamı aşağıdaki gibi yapılmıştır:

csecgin@csecgin-VirtualBox:~$ echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

Cassandra Key(kullanım için gerekli anahtar) ekliyoruz. Burada key yerine 111111111 yazılmıştır.

csecgin@csecgin-VirtualBox:~$ curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -sudo apt-key adv --keyserver pool.sks-keyservers.net --recv-key 111111111

Son olarak paket güncellenip, Apache Cassandra yüklenmiştir;

csecgin@csecgin-VirtualBox:~$ sudo apt-get update

csecgin@csecgin-VirtualBox:~$ sudo apt-get install cassandra

Apache Cassandra başlatılmıştır;

csecgin@csecgin-VirtualBox:~$ sudo systemctl start cassandra.service

Apache Cassandra durdurulmuştur;

csecgin@csecgin-VirtualBox:~$ sudo systemctl stop cassandra.service

Servisler, sistem önyüklemesinde etkin olmadığından aşağıdaki komut ile etkinleştirilmiştir:

csecgin@csecgin-VirtualBox:~$ sudo systemctl enable cassandra.service

Ubuntu 16.04 versiyonlu işletim sistemine yukarıdaki gibi Apache Cassandra kurulmuş. Tez için yapılacak testler için kullanılmıştır[url-4].

4.2 Cassandra Veri Tipleri(Data Type)

Cassandra'da veriler tablo satır ve sütun bazlı olarak tutulur. Her bir sütuna tanımlanacak bilecek veri tipleri Çizelge 4.1'de gösterilmiştir[url-5].

Çizelge 4.1: Cassandra Veri Tipleri

Veri Tipi	Açıklama
ascii	US-ASCII karakter dizesi tutar.
bigint	64-bit uzunluğunda eksi değerlerde alabilen, büyük tam sayılardır.
blob	Rastgele dizilen bytes, Hexadecimal olarak ifade edilir. Dosya depolamak için kullanılır.
boolean	True veya False değerler tutar.
counter	Dağıtık sayaç değeri alır(64-bit uzunluğunda long değer alabilir.)
decimal	Değer bazlı tam sayı tutar.
float	4 byte uzunluğunda 9 basamağa kadar sayı tutar.
int	4 byte uzunluğunda -2147483647 ile 2147483647 aralığında veri tutar.
list	Birden fazla değeri içerisinde barındırabilir.(diziler gibi)
map	JSON standartlarında veri tipidir. Örneğin; {baslik: yazi, gove: yazar...}
set	Birden fazla değeri içerisinde barındırabilir.(diziler gibi)
text	UTF-8 tipinde metin tutar.
timestamp	8 byte ile ifade edilir. Yıl, ay, gün, saat, dakika, saniye tutar.
uuid	32 hexadecimal basamak ve 4 adet tire kullanılarak yazılarak ID oluşturur.
varchar	UTF-8 tipinde metin tutar.
varint	Rastgele dizilmiş tam sayıları tutar.

4.3 Cassandra Nesneleri

Cassandra veri tabanında, veriler fiziksel olarak keyspace'lerde saklanır. Keyspace kavramı mantıksal olarak ilişkisel veri tabanlarındaki Database kavramına tekabül eder. Keyspaceleri oluştururken cluster ve replication opsiyonları ile oluşturulur. Bu sayede veriler düğümler üzerine dağıtılır. Cassandra'da büyük küçük harf duyarlılığı vardır. Bu nedenle küçük harflerle

oluşturulan bir keyspace yine küçük harfler kullanılarak çağrılmıştır. Büyük harflerle tanımlanan keyspace ve tablolar tırnak işareti ile çağrılmaktadır.

Veriler satır ve sütun bazlı tablolarda tutulur. CQL, Sql diline benzediğinden, Sql'deki bir DDL işlemi Cassandra ortamına uygulanabilir. Buradaki en önemli husus Cassandra'da oluşturulan bir tabloya primary key tanımlanmak zorundadır. Örnek tablo oluşturma kodu aşağıdaki gibidir:

```
CREATE TABLE test.calisanlar(
id uuid,
isim varchar,
soyisim varchar,
dogum_tarihi date,
PRIMARY KEY(id)
);
```

İlişkisel veri tabanları uzun yıllardır kullanıldığından Cassandra'daki nesneleri en kolay yol ile anlatmak için ilişkisel veri tabanlarındaki nesneler Çizelge 4.2'de karşılaştırılmıştır:

Çizelge 4.2: RDBMS ve Cassandra Nesneleri

RDBMS(MySQL)	**Cassandra**
Database	KeySpace
Table	Table
Row	Row
Column	Column

5. MONGODB

MongoDB, 2009 yılıda MongoDB Inc tarafından yayınlanmıştır. Döküman tabanlı ve açık kaynak bir VTYS'dir. Günümüzde en sık kullanılan NoSQL veritabanıdır. C++ programlama dili ile yazılmıştır. MongoDB verileri BSON(Binary JSON) formatında tutar. Bu format JSON veri formatına çok benzemektedir. Bu sayede pek çok farklı veri tiplerini desteklemektedir. MongoDB nesne tabanlı programlama prensiplerine çok yakın bir VTYS'dir. Windows, OS X, Linux, Solaris işletim sistemlerinde çalışabilir. Sharding metodu ile verileri bölümleme(partitioning) yapar. In-Memory kapasitesi yoktur. Kendine özgü bir sorgulama dili vardır. Bunun yanı sıra javascript kodlarını direkt çalıştırır. MongoDB'de tablo yerine collection, satır yerine doküman yer alır. MongoDB CAP(consistency, availability, partition Tolerance) teoreminden consistency ve partition tolerance'ı sağlar. Yani tüm istemciler(client) her zaman aynı veriyi görüntüler, bunun yanı sıra partition tolerans veriler başka sunuculara dağıtılarak yatak ölçekleme yapılır ve sisteme gelecek yükü dağıtır(yatay ölçekleme)[16].

5.1 MongoDB Kurulum

MongoDB Inc. Şirketi tarafından geliştirilen MongoDB VTYS kurulumu Ubuntu 16.04 versiyonlu işletim sistemine aşağıdaki şekilde kurulmuştur:

MongoDB repository kullanımı için ihtiyaç duyulan key aşağıdaki gibi tanımlanmıştır: Gerçek key yerine örnek olarak 11111 yazılmıştır.

csecgin@csecgin-VirtualBox:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 11111

csecgin@csecgin-VirtualBox:~$ # echo "deb http://repo.MongoDB.org/apt/ubuntu "$(lsb_release -sc)"/MongoDB-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/MongoDB-org-3.2.list

Ubuntunun yeni eklenen repository paketlerini okuyabilmesi için güncelleme aşağıdaki gibi yapılmıştır:

csecgin@csecgin-VirtualBox:~$ sudo apt-get update

MongoDB kurulumu için aşağıdaki komut çalıştırılmıştır:

csecgin@csecgin-VirtualBox:~$ sudo apt-get install –y MongoDB-org

MongoDB 3.6.4 versiyonunu yüklemek ve her bileşen paket sürümünün aynı olması için her bileşen paketi için bu versiyonu tanımlayan aşağıdaki kod çalıştırılmıştır:

csecgin@csecgin-VirtualBox:~$ sudo apt-get install -y MongoDB-org=3.6.4 MongoDB-org-server=3.6.4 MongoDB-org-shell=3.6.4 MongoDB-org-mongos=3.6.4 MongoDB-org-tools=3.6.4

MongoDB sırasıyla başlatma, kapatma, yeniden başlatma;

csecgin@csecgin-VirtualBox:~$ sudo service mongod start

csecgin@csecgin-VirtualBox:~$ sudo service mongod stop

csecgin@csecgin-VirtualBox:~$ sudo service mongod restart

MongoDB kullanmaya aşağıdaki şekilde başlanmıştır:

csecgin@csecgin-VirtualBox:~$ mongo 127.0.0.1:27017

Ubuntu 16.04 versiyonlu işletim sistemine yukarıdaki gibi MongoDB kurulmuş. Tez için yapılacak testler için kullanılmıştır[url-6].

5.2 MongoDB Veri Tipleri(Data Type)

MongoDB'de veri tipleri Çizelge 5.1'de gösterilmiştir[url-7].

Çizelge 5.1: MongoDB Veri Tipleri

String	Veri tutmak için en yaygın kullanılan tiptir. UTF-8 formatında olmalıdır.
Integer	Sayısal verileri tutmak için kullanılan veri tipidir. 32/64 bit olabilir
Boolean	Boolean(True/False) değerlerini tutan tiptir.
Double	Virgüllü sayıları tutmak için kullanılan veri tipidir.
Min / Max Keys	Minimum ve maksimum değeri karşılaştırmak için kullanılan veri tipidir.
Arrays	Array, List veye çoklu değerleri saklamak için kullanılan tiptir.
Timestamp	Zamanı refere eden tiptir. Bir doküman eklendiğinde veya değiştirildiğinde kaydedilir.
Object	Dökümanlar için kullanılan data tipidir.
Null	Null değerleri saklamak için kullanılan tiptir.
Date	Mevcut tarih ve saati UNIX tipinde tutmak için kullanılan tiptir.
ObjectID	Dökümanların ID lerini tutmak için kullanılan bir tiptir.
Binary Data	Binary veriyi tutan tiptir.

5.3 MongoDB Nesneleri

MongoDB veri tabanında, veriler fiziksel olarak Database'de saklanır. Database kavramı ilişkisel veri tabanlarındaki Database kavramı ile aynıdır. MongoDB'de ilişkisel veri tabanlarından farklı olarak şema(schema) olmaksızın veriler eklendikçe veya güncellendikçe veri tabanı nesneleri dinamik olarak oluşur.

MongoDB'de veriler collectionlar içerisinde döküman olarak tutulur. Collectionlar, ilişkisel veri tabanlarındaki tablolara karşılık gelmektedir. Collection yapısı içerisinde her bir kayda ait saklanacak veri başlıkları Field olarak adlandırılır. Collectionlar dinamik olarak oluşturulabildiğinden, Fieldlarda dinamik olarak eklenebilir veya çıkartılabilir. Collection oluşturmak için db.createCollection(collection_adı) komutunu kullanırız.

MongoDB'de document(döküman), collectionlara eklenecek fiziksel verileri temsil etmektedir. MongoDB ilişkisel veri tabanlarından farklı olarak bir Field içerisinde birden fazla bilgiyi tutabilmektedir. Örnek bir document aşağıdaki gibidir:

```
{
id: ObjectId(),
baslik: ' Lets Go'
by: {Fname: "Caner", lname: "Seçgin"},
like: 5,
comments: [ { fname:"Fatih", lname:"Geylani", text:"Ne haber"},
    { fname:"Furkan", lname:"Sengul", text:"ekle lütfen"}],
content: "Okullar kapandı."
}
```

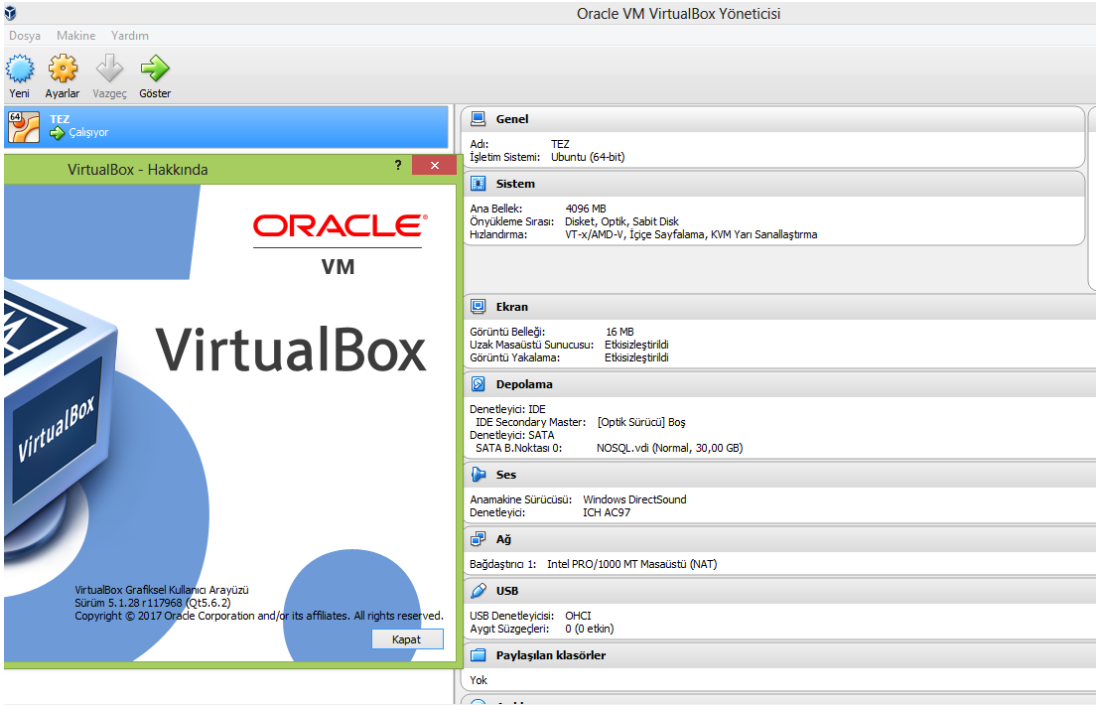
İlişkisel veri tabanları uzun yıllardır kullanıldığından MongoDB'deki nesneleri en kolay yol ile anlatmak için ilişkisel veri tabanlarındaki nesneler ile aşağıda Çizelge 5.2'de karşılaştırılmıştır:

Çizelge 5.2: RDBMS ve MongoDB Nesneleri

RDBMS(MySQL)	**MongoDB**
Database	Database
Table	Collection
Row	Document
Column	Field

6. ORACLE VIRTUALBOX

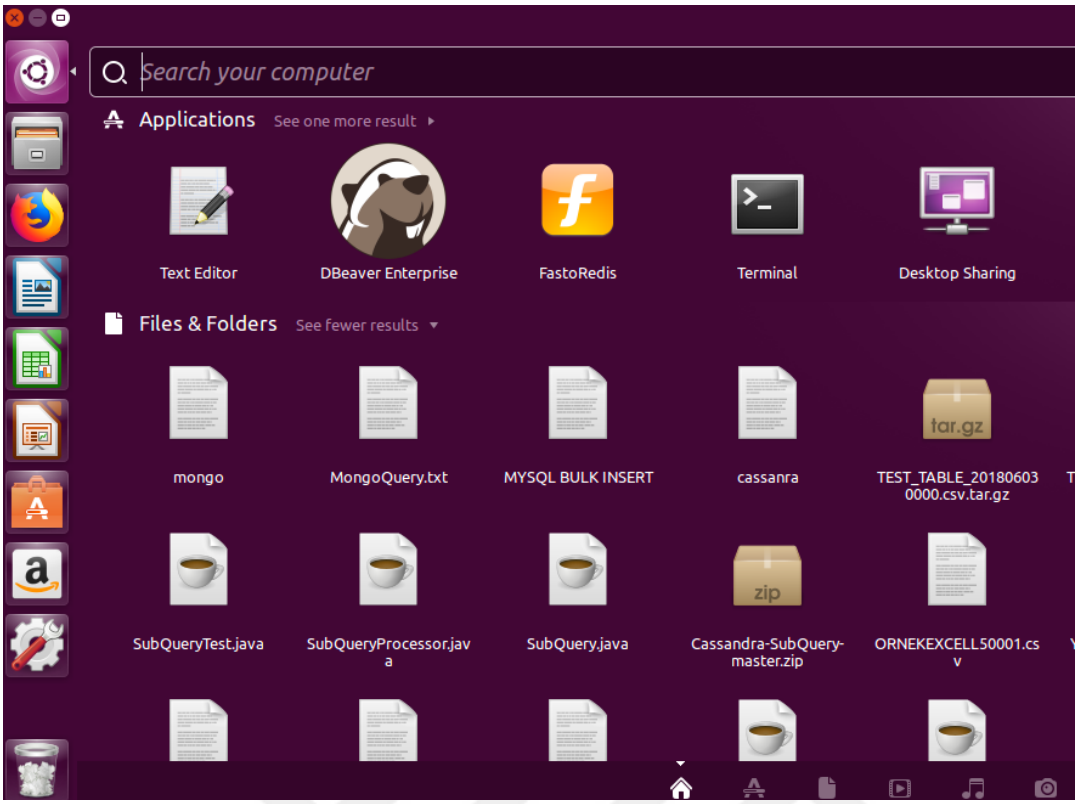
Sanal makine olarak Oracle VirtualBox 5.1.28 sürümü kullanıldı. Burada Linux çekirdeğine sahip Ubuntu 16.04 versiyonu yüklendi. VirtualBox çok platformlu bir sanallaştırma programıdır. Virtualbox Oracle firması tarafından ücretsiz sağlanmaktadır. Kurulum aşamasında Hyper-v özelliği açık olmalıdır. Sanal makinede çalışacak işletim sistemi için disk kalıbı yerleştirilmiş olmalıdır. Lokal bilgisayar üzerine bir harici disk ve bellek ayırıp veri tabanlarını sanal makine üzerinde kurulumu yapılarak. Mevcut işletim sisteminde çalışan programların yapılacak performans testlerine etkisi en aza indirgendi. Virtualbox bir çok işletim sistemi desteklemektedir. Bu işletim sistemleri; Microsoft Windows, Linux, Solaris, BSD, Mac OSX, IBM OS/2, DOS, Ubuntu gibi işletim sistemleridir. Kurulumda 4gb ram, 100gb hard disk ayrılmıştır. Makine üzerinde bulunan Intel(R) Core(TM) i7-3632QM CPU @2.20G işletim sistemi ile 2195.012 Mhz Cpu kullanılmıştır[url-8][url-9]. VirtualBox program görüntüsü Şekil 6.1'de gösterilmiştir.



Şekil 6.1: VirtualBox Başlangıç Ekranı

7. UBUNTU

Ubuntu, Linux işletim sistemi çekirdeği temel alınarak geliştirilen açık kaynak kodlu, ücretsiz bir işletim sistemidir. Ubuntu'nun ilk masaüstü sürümü 2004 yılında yayınlanmıştır. Ubuntu 2004 yılında Güney Afrikalı girişimci Mark Shuttleworth öncülüğünde Canonical Ltd. şirketi tarafından yayınlanmıştır. Günümüzde yine Mark Shuttleworth'ün sahibi olduğu Canonical Ltd. ve Ubuntu gönüllüleri topluluğu tarafından geliştirilmektedir. Ubuntu'nun her 6 ayda bir yeni sürümü yayınlanmaktadır. Ubuntu sözcüğü, Zulu dilinde "insanlık" anlamına gelir. Ubuntu, günümüzde on milyonlarca müsaüstü-dizüstü bilgisayarda, sunucu, dron, otomobil ve çok çeşitli nesnelerin interneti adı altındaki cihazda kullanılmaktadır. Ubuntu her türlü cihazda ücretsiz bir şekilde lisanslanmakta ve kullanılmaktadır. Kodlar herkese açık bir şekilde paylaşılmaktadır. Canonical şirketi Ubuntu'nun her zaman ücretsiz olacağının taahhütünü vermektedir. Ubuntu bünyesinde geliştirilen kodlar GNU Genel Kullanım Lisansı ile lisanslandığı için Ubuntu'nun tüm kodları kamu malı gibi herkese aittir. Canonical, Ubuntu'nun marka haklarına sahiptir. Bu şirket tarafından Ubuntu lisansı ücretlendirmeye çalışılsa dahi, kişi veya topluluklar tarafından bu kodlar tekrar derlenerek başka bir isim ile bu işletim sistemi tekrar yayınlanabilir. Bu şirket gelirlerini lisans satışları yerine kurumsal ve profesyonel kullanım için verdiği teknik destek ile sağlamaktadır. Ubuntu işletim sistemi Şekil 7.1'de gösterilmiştir.



Şekil 7.1: Ubuntu MasaÜstü

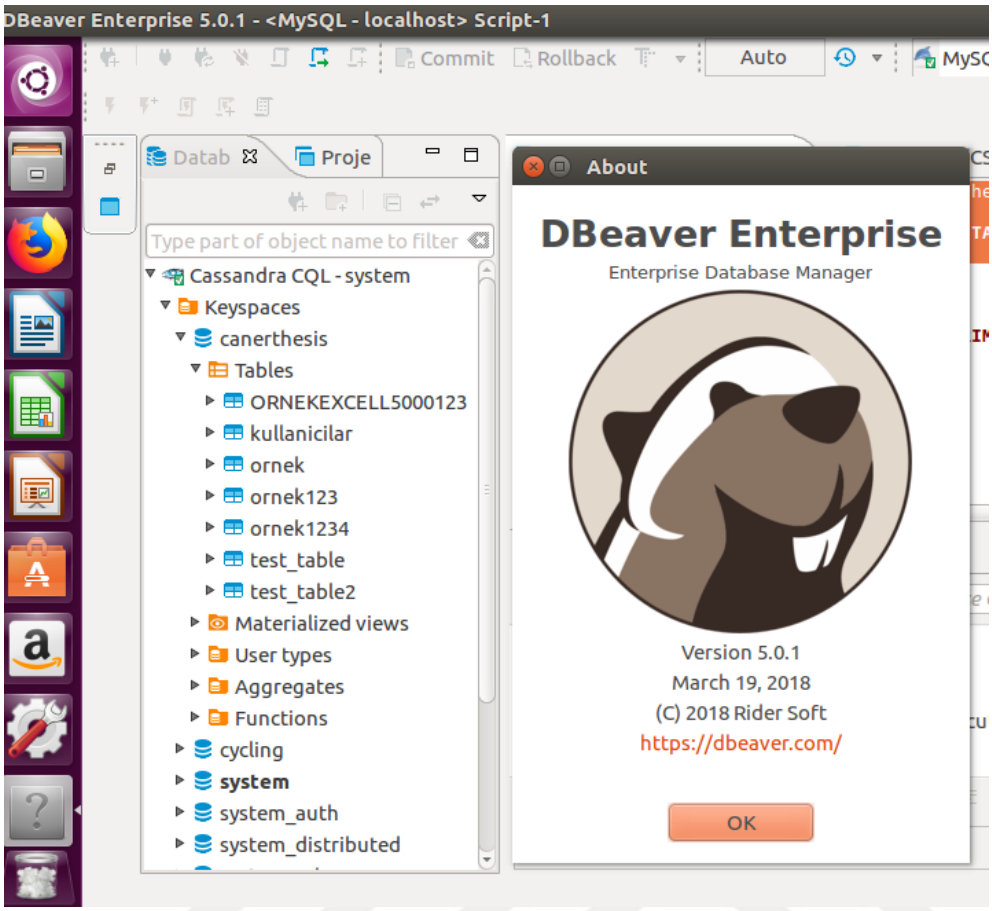
Ubuntu, dünya çapında milyonlarca kişi tarafından kullanılan ücretsiz LibreOffice paketini içermektedir. Bu paket sayesinde Microsoft Office ile hazırlanmış belgeleri açılabilir ve düzenlenebilir. LibreOffice, The Document Foundation(Belge Vakfı) öncülüğünde geliştirilmekte olup Microsoft Office ile büyük oranda uyumluluk göstermekte ve Windows işletim isteminde de çalışabilmektedir.

Ubuntu, pek çok programlama dilini ve geliştirme ortamını destekler. Ubuntu üzerinde kullanılabilen programlar; C, C++, Objective-C, Java, Python, Perl, Ruby, Pascal, C#. Bunun yanı sıra bir çek VTYS'ini desteklemektedir. Bu çalışmada kullanılan MongoDB, MySQL ve Cassandra VTYS'lerini desteklemektedir.

Bu çalışmada Ubuntu 16.04 versiyonu kullanılmıştır. Bu çalışmada kullanılmasının amacı; işletim sisteminin ve uygulamaların çok pratik bir şekilde kurulumu ve yönetimidir[url-10].

8. DBEAVER

DBeaver, geliştiriciler ve veri tabanı yöneticileri için geliştirilmiş, açık kaynak kodlu ve community edition(topluluk sürümü) için ücretsiz bir veri tabanı yönetim aracıdır. Dbeaver üzerinde SQL sorguları, düzenlemeleri, transformasyonlar, objelerin ağaç yapısı, log sistem yönetimi, veri tabanı bağlantı oturum(session) izlemeleri, veri tabanı yöneticisi işlemlerin tamamlanması gibi işlevleri vardır. Teknik olarak bakılacak olursa; özelleştirilmiş kullanıcı ara yüzü, çoklu platform desteği, çeşitli eklenti yazabilme(plugins), JDBC sürücüsüne(driver) sahip tüm veri tabanı yönetim sistemleri desteği, herhangi bir harici(external) veri kaynağını(odbc,oledb vb..) JDBC sürücüsüne sahip veya sahip olmaması farketmeksizin işleme yeteneği gibi özellikleri vardır. Veri transferi için CSV, HTML, XML, XLSX, XLS gibi formatları desteklemektedir. Dışarı veri aktarma(export) işlemlerinde kendi ara yüzü içinde tanımlı bağlantılar içerisindeki veri tabanlarına tablo oluşturup, verilerin aktarımını yapmaktadır. Meta data(ana veri) yönetimi için tablolar, viewlar, kolonlar, indeksler, prosedürler, triggerlar, tablespace ve partitions, kullanıcı ve rol yetkilendirmesi yapabilmektedir. DDL desteği olarak SQL92 desteklemektedir. Veri tabanları olarak; MySQL, MongoDB, Apache Cassandra, MariaDB, PostgreSQL, GreenPlum, Oracle,DB2, SQL Server, Informix, Sysbase, SAP MaxDB, FireBird, Ingres, Linter, MimerSQL, CUBRID, Exasol, Vertica, Tedata, SAP Hana, Netezza, PrestoDB, ClickHouse, Redis, NuoDB, RedShift, Snowflake, Apache Hive, SparkHive, Apache Phonix, Apache Impala, Neo4j, OrientDB, SQLite, H2 gibi veri tabanlarını desteklemektedir[url-11][url-12][url-13].Dbeaver programı arayüzü Şekil 8.1'de gösterilmiştir.



Şekil 8.1: DBeaver Arayüzü

Bu çalışmada DBeaver kullanılmasının amacı; çalışmaya konu olan veri tabanı yönetim sistemlerinin hepsine destek vermesi ve SQL dili kullanılarak NoSQL veri tabanlarında sorgulama ve veri manipülasyonu yapılabilmesidir. Bu akademik çalışmayı desteklemek için DBeaver kurumsal lisansı ücretsiz olarak temin etmiştir.

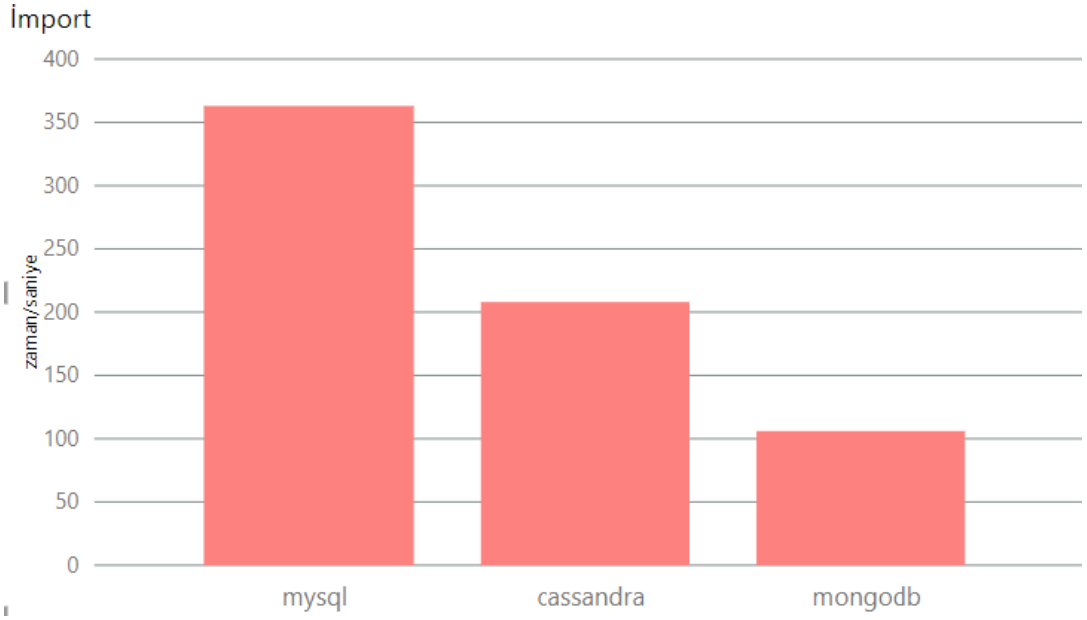
9. TESTLER

Testler, Virtualbox ile sanal makine içerisinde sadece test edilecek veri tabanlarının ve derleyicinin kurulması ile laboratuvar ortamını oluşturularak başka faktörlerden en az etkilenecek şekilde yapılmıştır. Testler sırasında teste konu olan veri tabanı dışındaki tüm veri tabanı ve işlemler kapatılmıştır.

Tez çalışmaları için 20 alanı olan 1 milyon kaydın olduğu bir veri seti oluşturuldu ve bu veri seti veri tabanlarına yüklenerek testler yapıldı. Test sonuçları grafikler ile gösterilmiştir.

9.1 Verilerin Sisteme Yüklenmesi

Önceden hazırlanmış ve testlere uygun olduğu düşünülen veri seti Ubuntu üzerine csv uzantılı bir dosya olarak koyulmuş, buradan direkt veri tabanlarının içerisine aktarılmıştır. Csv içerisinde 1 milyon kayıt bulunmaktadır. Bu kayıtların disk üzerinde kapladığı yer 150 megabyte'dır. 1 milyon kaydı sisteme 106 saniyede yükleyerek(import) en hızlı yüklemeyi yapan MongoDB oldu. Performans sonuçları Şekil 9.1'de gösterilmiştir.



Şekil 9.1: Verilerin sisteme yüklenmesi

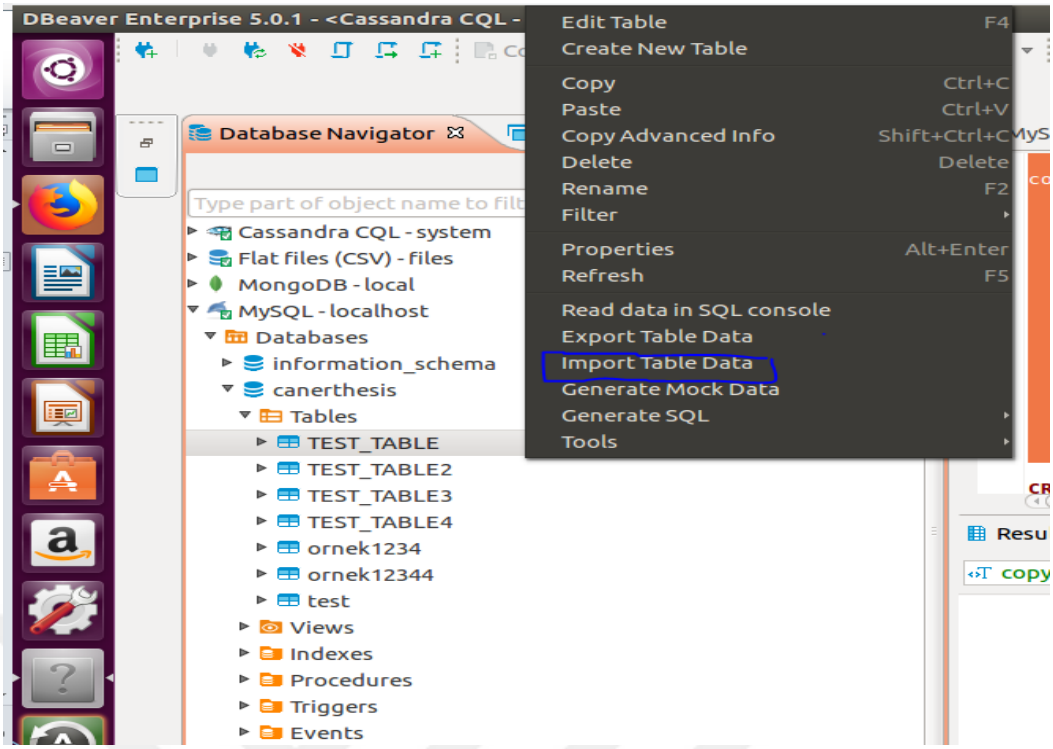
Cassandra Veri tabanına import ederken aşağıdaki yöntem ile veri seti içeriye atılmıştır. Bu kod ubuntu işletim sistemi üzerindeki terminalden çalıştırılmıştır.

```
csecgin@csecgin-VirtualBox:~$ cqlsh

Connected to Test Cluster at 127.0.0.1:9042.

[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]

Use HELP for help.

cqlsh> copy canerthesis."test_table"("sira", "bill_acct_dt", "bill_acct_dt1",
"bill_id", "bill_id1", "city", "city1", "phase", "phase1", "t0", "t01", "t1", "t11",
"t2", "t21", "t3", "t31" ) from
'/home/csecgin/Desktop/files/TEST_TABLE_201805261657.csv'
```

MongoDB Veri tabanına import ederken aşağıdaki yöntem ile veri seti içeriye atılmıştır. Bu kod ubuntu işletim sistemi üzerindeki terminalden çalıştırılmıştır.

```
csecgin@csecgin-VirtualBox:~$ sudo systemctl start mongod

csecgin@csecgin-VirtualBox:~$ mongoimport --host=127.0.0.1 -d canerthesis -
c test_table --type csv --file
/home/csecgin/Desktop/files/TEST_TABLE_201805261657.csv –headerline
```

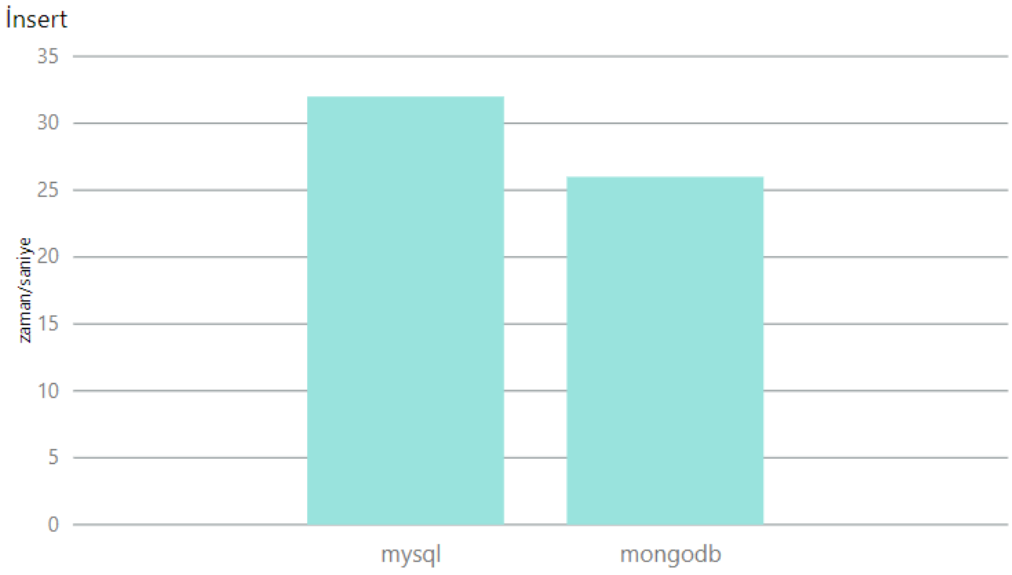
MySQL Veri tabanına import ederken DBever'ın sunduğu import yöntemi kullanılmıştır. DBever import yöntemi Şekil 9.2'de gösterilmiştir.



Şekil 9.2: DBeaver Import Yöntemi

9.2 Toplu Okuma ve Yazma Testi

Toplu(bulk) veri yazma(insert) işleminde cassandra desteklemediği için dahil edilmemiştir. Bu testte 1 milyon kaydı okuma ve yazma yapılmıştır. Bu test sonucunda mongo db 26 saniyede MySQL veri tabanı ise 32 saniyede işlemi tamamlamıştır. Performans sonuçları Şekil 9.3'de gösterilmiştir.



Şekil 9.3: Toplu okuma ve yazma testi

Mongo veri tabanına aşağıdaki yöntem ile test_table collection'undaki tüm veriler test_table2 collection'una atılmıştır. Tüm collection'u atıldığından Field'ları tek tek yazma gereksinimi duyulmamıştır. Kodlar Dbeaver aracılığı ile çalıştırılmıştır.

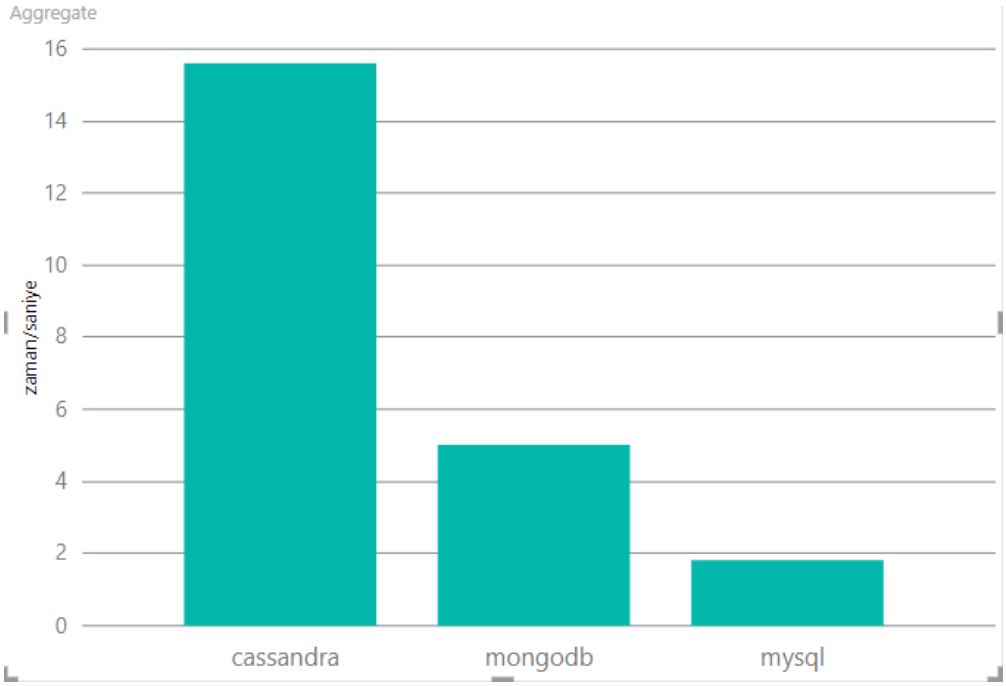
```
var bulkIns = db.test_table2.initializeUnorderedBulkOp()

db.test_table.find({}).forEach(function(doc){bulkIns.insert(doc);})

bulkIns.execute()
```

MySQL veri tabanına aşağıdaki yöntem ile test_table tablosundaki tüm veriler test_table2 tablosuna atılmıştır. Tüm tablo atıldığından column'ları tek tek yazma gereksinimi duyulmamıştır. Kodlar Dbeaver aracılığı ile çalıştırılmıştır.

```
INSERT INTO canerthesis.TEST_TABLE2

SELECT * FROM canerthesis.TEST_TABLE
```

9.3 Toplam ve Ortalama Alma İşlemi

Bu testte 1 milyon kayıt içerisindeki 3 sayı(numeric) alanın hem ortalaması(avg) hem de toplamı(sum) alındı. MySQL veri tabanı 1.8 saniye ile bu işlemi en hızlı yapan veri tabanı oldu. Performans sonuçları Şekil 9.4'de gösterilmiştir.



Şekil 9.4: Toplam ve Ortalama işlemi sonuçları

Cassandra veri tabanında aggregation(grup) işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
select sum(t1),sum(t2),sum(t3),avg(t1),avg(t2),avg(t3) from test_table2
```

MongoDB veri tabanında aggregation(grup) işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
db.test_table2.aggregate(
 [{
  "$group": {
   "_id": null,
   "AVG(t1)": {
    "$avg": "$t1"
   },
   "AVG(t2)": {
    "$avg": "$t2"
   },
   "AVG(t3)": {
    "$avg": "$t3"
   },
   "SUM(T1)": {
    "$sum": "$T1"
   },
   "SUM(T2)": {
    "$sum": "$T2"
   },
   "SUM(T3)": {
    "$sum": "$T3"
   }
```

}

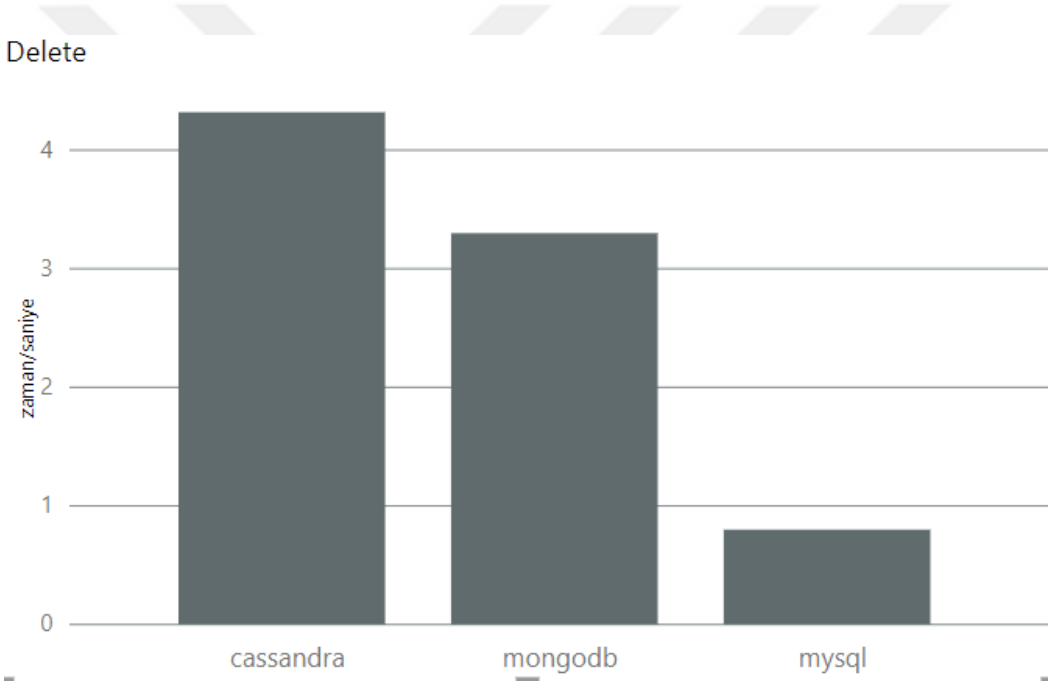
}])

MySQL veri tabanında aggregation(grup) işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
select sum(t1),sum(t2),sum(t3),avg(t1),avg(t2),avg(t3) from test_table2
```

9.4 Veri Silme İşlemi

1milyonluk tablodan 20 bin kaydı silme işlemini test ettik. MySQL veri tabanı 0.8 saniye ile bu işlemi en hızlı yapan veri tabanı yönetim sistemi oldu. Performans sonuçları Şekil 9.5'de gösterilmiştir.



Şekil 9.5: Veri Silme İşlemi

Cassandra veri tabanında veri silmek veya güncellemek istendiğinde ilgili kayıtlara gidildiğin-de mutlaka birincil anahtar(pk) kullanılmalıdır. Bu sebeple 20000 kayıt Cassandra ortamından silinirken birincil anahtar where koşulunda belirtilmiştir.

```
delete from test_table2

where  sira in(

1    ,2    ,3    )
```

MongoDB veri tabanında veri silme işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

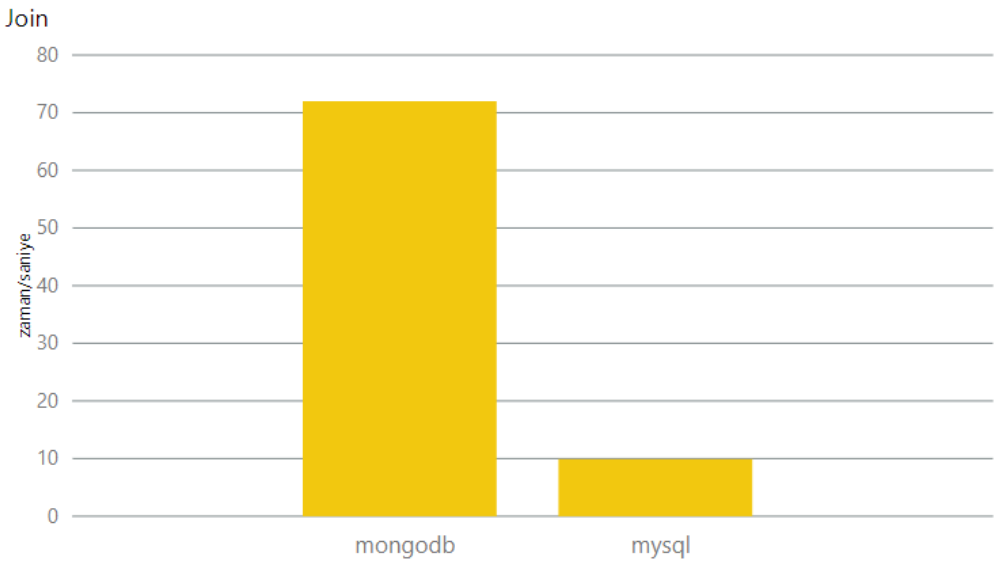
```
db.test_table2.remove({"CITY": "Tokyo"})
```

MySQL veri tabanında veri silme işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
delete from test_table2

where  CITY= 'Tokyo'
```

9.5 Join İşlemi

Veri tabanında tabloları birbirleriyle ilişkili oldukları alanlar, ID'ler üzerinden birleştirerek(join) mevcut veri seti genişletilerek birleştirilir(denormalize). NoSQL veri tabanlarında genel olarak tablolar denormalize dizayn edilir ve tablolar arasında ilişki tutulmaz. Bu sebeple Cassandra bu teste dahil edilmedi. MongoDB'de ise referans doküman ile ilgili ilişki bir dizi içerisinde tutulur; fakat bu da çok performanssız olduğundan bu testte MongoDB'deki aggregate fonksiyonundan faydalanıldı. 2 adet 1 milyon kaydın olduğu tabloyu birleştirdiğimizde. MySQL veri tabanı MongoDB'ye oranla 7 kat bu birleştirmeyi daha hızlı yaptı. Performans sonuçları Şekil 9.6'da gösterilmiştir.



Şekil 9.6: Join İşlemi

MongoDB veri tabanında join işlemleri aşağıdaki şekilde yapılmaktadır: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

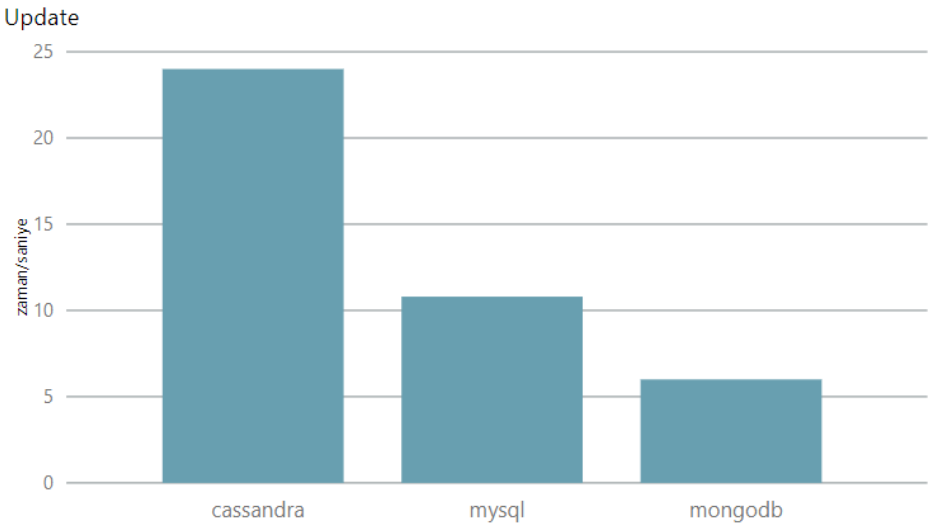
```
db.test_table2.aggregate([
{
$lookup: {
from: "test_table3",
localField: "SIRA",
foreignField: "SIRA",
as: "embeded_data"
}
}
]);
```

MySQL veri tabanında join işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
SELECT COUNT(*) FROM canerthesis.TEST_TABLE2 t1
LEFT JOIN canerthesis.TEST_TABLE3 t2 on t1.SIRA=t2.SIRA
```

9.6 Veri Güncelleme İşlemi

1 milyon tablo içerisinde 122660 adet kaydı güncelleme(update) işlemi test edilmiştir. Bu testlerde MongoDB 6 saniye gibi bir sürede işlemi bitirerek en hızlı sonucu elde etmiştir. Performans sonuçları Şekil 9.7'de gösterilmiştir.



Şekil 9.7: Veri Güncelleme İşlemi

Cassandra veri tabanında veri güncellemek istendiğinde ilgili kayıtlara gidildiğinde mutlaka birincil anahtar(pk) kullanılmalıdır. Bu sebeple kayıtlar Cassandra ortamında güncellenirken birincil anahtar where koşulunda belirtilmiştir. Güncellenecek tüm kayıtlar için bu işlemi göstermek zor olacağından aşağıdaki gibi birkaç kayıt için örnek güncelleme işlemi gösterilmiştir: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
update test_table2

set t3= ‘100’

where  sira in(

1   ,2   ,3   )
```

MongoDB veri tabanında veri güncelleme işlemleri aşağıdaki şekilde yapılmaktadır. Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
var bulkUp2 = db.test_table2.initializeOrderedBulkOp()

bulkUp2.find({"CITY": "Tokyo"}).update({ $set: { T1: 100 } })

bulkUp2.execute()
```

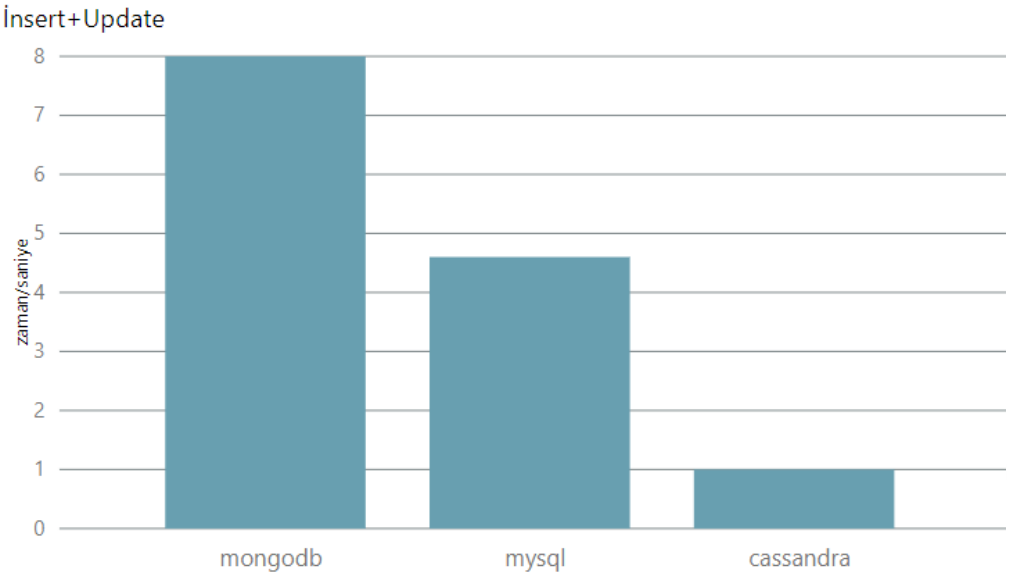
MySQL veri tabanında veri güncelleme işlemleri aşağıdaki şekilde yapılmaktadır: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
update test_table2

set t3= ‘100’
```

where city=’Tokyo’

9.7 Sırayla Veri Ekleme, Güncelleme

Günlük hayatta kullanılan veri tabanlarında sisteme anlık olarak işlemler(transaction) gelir. Bu şekilde çalışan bir sistem düşünülerek 1000 farklı insert 1000 farklı update işlemi veri tabanlarına gönderildi. Cassandra bu sıralı işlemlerde 1 saniye gibi hızlı bir süre ile en iyi performansı sağladı. Performans sonuçları 9.8’de gösterilmiştir.



Şekil 9.8: Sırayla Veri Ekleme, Güncelleme

Cassandra ortamında insert ve update kodu aşağıdaki şekilde yapılmıştır: Güncellenecek tüm kayıtlar için bu işlemi göstermek zor olacağından aşağıdaki gibi birkaç kayıt için örnek güncelleme işlemi gösterilmiştir: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

INSERT INTO canerthesis. testcaner(ad,id,t1,t2,t3) VALUES('caner', 1, '100', '150', '200');

UPDATE canerthesis. testcaner set t1= '200' where id= 1;
INSERT INTO canerthesis. testcaner(ad,id,t1,t2,t3) VALUES('caner', 2, '100', '150', '200');

UPDATE canerthesis. testcaner set t1= '200' where id= 2;

MongoDB ortamında insert ve update kodu aşağıdaki şekilde yapılmıştır: Güncellenecek tüm kayıtlar için bu işlemi göstermek zor olacağından aşağıdaki gibi birkaç kayıt için örnek güncelleme işlemi gösterilmiştir: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
db.testcaner.insert({ad:"caner", t0:1,t1: 100, t2:150,t3: 200})
```

```
db.testcaner.update({t0:1},{t1:150})
```

MySQL ortamında insert ve update kodu aşağıdaki şekilde yapılmıştır: Güncellenecek tüm kayıtlar için bu işlemi göstermek zor olacağından aşağıdaki gibi birkaç kayıt için örnek güncelleme işlemi gösterilmiştir: Bu kod DBeaver aracılığı ile çalıştırılmıştır.

```
INSERT INTO canerthesis. testcaner(ad,id,t1,t2,t3) VALUES('caner', 1, '100', '150', '200');
```

```
UPDATE canerthesis. testcaner set t1= '200' where id=  1;
```

```
INSERT INTO canerthesis. testcaner(ad,id,t1,t2,t3) VALUES('caner', 2, '100', '150', '200');
```

```
UPDATE canerthesis. testcaner set t1= '200' where id=  2;
```

10. SONUÇ

Unutulmamalıdır ki; CAP teoremi güncelliğini korumakta. Dolayısıyla bir veri tabanı yönetim sistemi aynı zamanda consistency, availability ve partition tolerance'ı sağlayamıyor. Test edilen VTYS'lerden MySQL hem tip olarak ilişkisel veri tabanı hem de consistency ve availability sağlar. Cassandra, availability ve partition tolerance sağlayan sütun bazlı bir NoSQL veri tabanıdır. MongoDB, consistency ve partition tolerance sağlayan doküman tabanlı bir NoSQL veri tabanıdır. Bu sayede hem farklı tip veri tabanlarını, hem de farklı amaçlara hizmet eden veri tabanlarını incelemiş olduk. Cassandra, consistency(tutarlılık) sağlamayı garanti edemediği için banka gibi verisi çok önemli sistemlerde kullanılmaması öneriliyor. Bunun sebebi tüm kullanıcılar, istemciler her zaman aynı veriyi göremeyebiliyor. Zaman içerisinde veriler eşitlense de o an için görülemiyor. MongoDb consistency(tutarlılık) sağlıyor; fakat sürekli availability(uygunluk/erişebilirlik) sağlayamıyor. Bu sebeple yine banka gibi sistemlerde herhangi bir anda sisteme erişilemezse çok ciddi sorunlar ortaya çıkabiliyor. MySQL ise hem sürekli erişim hem sürekli tutarlılık sağladığı için banka gibi çok önemli sistemlerde kullanılması mantıklı olacaktır; fakat MySQL gibi ilişkisel veri tabanlarında partition tolerance(bölümleme) yatay ölçekleme yapılamadığı için ciddi performans ve maliyet sorunları gözükmekte. Mevcut sistemimizin kaynakları yetersiz kalıyorsa sisteme ekleme yapamıyorsunuz. Mecburen mevcut sisteminizi kaynakları daha güçlü bir sisteme taşımak zorundasınız. Yapılan performans testlerinde toplu veri okuma/yazma ve toplu veri güncelleme işlemlerinde MongoDB çok performanslı sonuçlar vermiştir. Verileri toplu bir şekilde gruplayarak toplamak ve ortalamalarını almak, tabloları birleştirmek(join) ve silme işlemleri de en iyi sonuçları MySQL veri tabanı vermiştir. Mevcut çalışan bir sistem içerisinde anlık ayrı ayrı gelen veri yazma ve veri güncelleme işlemlerini en hızlı Cassandra yapmıştır.

Sonraki çalışmalarda NoSQL veri tabanlarının yatay ölçeklenebilir(partition tolerance) özelliklerinin performans testleri yapılabilir.

KAYNAKLAR

[1] **Cornelia Gyorodi,Rober Gyorodi, George Pecherle, Andrada Olah,** A Comparative Study: MongoDB vs. MySQL

[2] **CAN RAZBONYALI,** Big Data(Büyük Veri) Ve Geleneksel Veri Saklama Ve İşleme Yöntemlerine Etkisi Üzerine Bir Araştırma

[3] **Yılmaz GÖKŞEN, Hakan AŞAN**(2015) Veri Büyüklüklerinin Veri tabanı Yönetim Sistemlerinde Meydana Getirdiği Değişim: NoSQL.

[4] **Süleyman Eken, Fidan Kaya, Ahmet Sayar, Adnan Kavak**(2014)Döküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB yatay ölçeklenebilirlik karşılaştırması

[5] **Serdar ÖZTÜRK, Hatice Ediz ATMACA** 2017İlişkisel ve İlişkisel Olmayan(NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi

[6] **Christoforos Hadjigeorgiou**(2013) RDBMS vs NoSQL: Performance and Scaling Comparison

[7] **Kerem Kayabay, Mert Onuralp Gökalp, Mehmet Ali Akyol, P. Erhan Eren, Altan Koçyiğit**, Geleceğin kuruluşları için büyük veri: Mevcut durum ve eğilimler

[8] **Turgut Özseven,** Veri Tabanı Yönetim Sistemleri

[9] **Eric Brewer**(2012), CAP twelve,Years later: How the "Rules" have Changed

[10] **N. Lynch, S. Gilbert,** "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, vol. 33, no. 2, pp. 51-59 , 2002.

[11] **Talip Hakan Öztürk,** Oracle Database 11g R2

[12] **Murat Azimli**, Hiyerarşik, İlişkisel ve NoSQL Veritabanları

[13] L. Yishan, S. Manoharan, "A performance comparison of Sql and and NoSQL databases"

[14] **A. Lakshman, M. Avinash,"Cassandra: A** Decentralized Structured Storage System"

[15] **Wang,G**.(2012). The NoSQL Principles and Basic Application of Cassandra Model. International Conference on Computer Science & Service System

[16] **Nazım Emre Şavklı,** MongoDB

İNTERNET KAYNAKLARI

[url1] https://medium.com/cloud-and-servers/acid-nedir-53f729f2bbb2

[url-2] Açık Kaynak Kodlu Veritabanların Yükselişi, https://tr.linkedin.com/pulse/a%C3%A7%C4%B1k-kaynak-kodlu-veritabanlar%C4%B1n-y%C3%BCkseli%C5%9Fi-turan-bahattin-%C3%B6zen-msc

[url-3] https://db-engines.com/en/ranking

[url-4] https://www.rosehosting.com/blog/how-to-install-apache-cassandra-on-buntu-16-04/

[url-5] https://medium.com/@bsekerciler/apache-cassandra-nedir-nasıl-kurulu-ve-nasıl-kullanılır-b36a5e27e2f8
[url-6] https://www.rosehosting.com/blog/how-to-install-MongoDB-on-ubuntu-16-04/
[url-7] https://www.tutorialspoint.com/MongoDB/MongoDB_datatype.htm
[url-8] https://www.virtualbox.org/wiki/End-user_documentation
[url-9] https://www.virtualbox.org/manual/ch01.html#virtintro
[url-10] https://wiki.ubuntu-tr.net/index.php?title=Ubuntu_nedir%3F
[url-11] https://github.com/dbeaver/dbeaver/wiki
[url-12] https://dbeaver.com/faq/
[url-13] https://dbeaver.com/databases/

ÖZGEÇMİŞ

Caner SEÇGİN

Bahçelievler Mah. Çayır Sok.
+90 507 796 97 80
No:5 D:26 Bahçelievler
secgincaner@gmail.com

Uzmanlik Alanlari:

- Sql Geliştirme Deneyimi(8 yıl)
- ETL Geliştirme Uzmanlığı(6 yıl)
- Veri Ambarı Tecrübesi(6 yıl)
- Oracle Veri Tabanı Yönetim Tecrübesi(3 yıl)
- C# ile Yazılım Geliştirme Tecrübesi(3 yıl)
- Microsoft SQL Server Tecrübesi(1 yıl)
- Veri Madenciliği Tecrübesi(1 yıl)

Eğitim Bilgileri:

- Şehremini Lisesi(2004-2007)
- İstanbul Aydin Üniversitesi
  Computer Engineering 2008-2013

Kurs Ve Sertifikalar:

- Oracle Database 11g: Introduction to SQL, 2013(Certificate)
- Oracle Database 11g: Administration Workshop I ,2013(Certificate:262711886)
- Oracle Database 11g: Administration Workshop II, 2013
- Advance SQL Tuning in Oracle Database 11g,2014
- MS SQL Server SQL Development : Bilge Adam Eğitim Merkezi, 2011
- Net Software Development : Bilge Adam Eğitim Merkezi, 2011

İş Deneyimi:

Garanti Teknoloji Şirketinde Kıdemli Veri Ambarı Uzmanı(06.2018 -)

- Veri Ambarı & Veri Tabanı Tasarımı
- Kaynak sistemlerden Veri Ambarına Modelleme
- Odi 12c ve Odi 10g ile Etl Geliştirme
- Test Süreçleri
- Kaynak Sistem Analizi

- Oracle Bi ve Sap BO ile Rapor Geliştirmeleri

Clk Enerji Şirketinde Kıdemli Veri Ambarı Uzmanı(11.2016 - 06.2018)

- Veri Ambarı & Veri Tabanı Tasarımı
- Kaynak sistemlerden Veri Ambarına Modelleme
- Odi 12c ve Odi 10g ile Etl Geliştirme ve Tüm Operasyonlar
- Test Süreçleri
- Kaynak Sistem Analizi
- Anlık Raporlama İhtiyacının Karşılanması
- Oracle Golden Gate ile Operasyonel Raporlama Sistemi Geliştirme
- Managing databases(Tablespaces, Memory Management, Disk Management, Monitoring, Object Privileges, Redolog files), Oracle DB 12c

Tani-Koç Grup Şirketinde Veri Ambarı Uzmanı (06.2015 – 11.2016)

- TANIDWH&TANI2 Veri Ambarı Projesinin Geliştirilmesi
- Kaynak sistemlerin Veri Ambarına göre Modellenmesi
- Odi 11g ve PL-SQL ile Etl kodlarının geliştirilmesi
- Test Süreçleri
- Kaynak Sistem Analizi
- Geliştirme ve Üretim ortamlarındaki Veri Tabanlarının Yönetimi

Ibm Şirketinde Kıdemli Veri Ambarı Uzmanı(10.2013 - 06.2015)

- Veri Ambarı & Veri Tabanı Tasarımı
- Kaynak sistemlerden cLDM'e Modelleme
- Odi 12c ve Odi 10g ile Etl Geliştirme ve Tüm Operasyonlar
- Test Süreçleri
- Kaynak Sistem Analizi
- Anlık Raporlama İhtiyacının Karşılanması
- Geliştirme ve Üretim ortamlarındaki Veri Tabanlarının Yönetimi

TAV Şirketinde .NET ve Veri Tabanı Uzman Yardımcısı(06.2011 - 09.2011)

- Otomatik Park Yönetim sisteminde C# ile Geliştirme ve Test Süreçleri

Projeler:

- Clk Enerji 4x4 EBS Projesinin Veri Ambarı ve Raporlama Projesi
- Opet Çağrı Merkezi için Veri Modeli ve Raporlama Projesi
- Ford Qlickview Projesi
- Vodafone Fixed Veri Ambarı Projesi
- Oracle Veri Tabanında Backup ve Recovery İşlemleri
- Oracle Veri Tabanı Network ve Memory Konfigrasyonları
- "www.secginmarket.com" Web Sitesi
- "www.ba-techs.com" Web Sitesi
- Decision Tree yöntemi ile Kanser hücrelerinin Analizi
- Şifreli Mesajlaşma Uygulaması

Yetenekler:

- Veri Tabanı: Oracle Database Veri Tabanı Yönetimi(Mimari, Memory, Güvenlik, Backup ve Recovery, Dİsk Yönetimi, Performans ve Sql İyileştirmeleri)
- SQL Geliştirilen Veri Tabanları: Oracle, MS SQL, MySQL
- Programlama Bilgisi: C#, C, Java
- Araçlar: Golden Gate, ODI 11g-12c, OBI 11g, Red Hat Linux 5.7, Oracle 11gR2, Oracle 12c db, MS SQL Server, TOAD , Rapid Miner, Visual Studio 2010, Netbeans, Dev C++, Protege, PowerDesigner
- Yabancı Dil: Upper Intermediate in English