T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

VERİ TABANLARINDA KULLANILAN METİN ARAMA YÖNTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

MAHAMOUD BRAHIM ADOUM

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Dr. Öğr. Üyesi Altan MESUT

EDİRNE-2022

Mahamoud BRAHIM ADOUM'un hazırladığı "**Veri Tabanlarında Kullanılan Metin Arama Yöntemlerinin Performans Karşılaştırması**" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından **Bilgisayar Mühendisliği** Anabilim Dalında bir **Yüksek Lisans** tezi olarak kabul edilmiştir.

Jüri Üyeleri: İmza

Dr. Öğr. Üyesi Altan MESUT .....

Dr. Öğr. Üyesi Emir ÖZTÜRK .....

Dr. Öğr. Üyesi Halil Nusret BULUŞ .....

Tez Savunma Tarihi: 22 / 06 / 2022

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Dr. Öğr. Üyesi Altan MESUT
Tez Danışmanı

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Hüseyin Rıza Ferhat KARABULUT
Fen Bilimleri Enstitüsü Müdürü

.....

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

22 / 06 / 2022

Mahamoud BRAHIM ADOUM

İmza

Yüksek Lisans Tezi

Veri Tabanlarında Kullanılan Metin Arama Yöntemlerinin Performans Karşılaştırması

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bu tezde, İlişkisel ve NoSQL veri tabanlarındaki genellikle metin indeksi adı verilen özel indeksler ile gerçekleştirilen Tam Metin Arama (FTS: Full Text Search) yöntemlerinin incelenmesi ve performans karşılaştırması yapılmıştır. Metin indekslerinde, kelimeler veya terimler bulundukları belgeler ile eşleştirilir ve gerekirse kelimelerin belgelerde kaç kez ve hangi konumlarda yer aldığına dair bilgiler de saklanabilir. Veri tabanında bir kelime veya terim arandığında, tüm dokümanları taramak yerine metin indeksleri kullanılarak arama işlemi çok daha hızlı yapılır. Son yıllarda birçok veri tabanı yönetim sistemi tam metin arama desteği sunmaya başlamış ve özellikle bu amaç için kullanılan Elasticsearch gibi arama motorları da ortaya çıkmıştır.

Tezin performans karşılaştırması bölümünde MSSQL Server, MySQL, MongoDB ve Elasticsearch veri tabanlarına makale özeti gibi küçük boyutlu ve kitap gibi büyük boyutlu çok sayıda metin verisi eklenmiştir. İlgili yöntemler ile metin indeksleri oluşturulduktan sonra test için belirlenen kelimeler bu indeksler üzerinde aranıp süre sonuçları elde edilmiştir. Aynı kelimeler Regex/Like türünde metin arama sorguları ile de aranmış ve sonuçlar indeks kullanılarak elde edilen sonuçlarla karşılaştırılmıştır.

Çalışmanın sonucunda ise hangi veri tabanı yönetim sistemi üzerinde hangi Tam Metin Arama veya Regex/Like yönteminin daha performanslı bir şekilde çalıştığı ortaya çıkarılmıştır.

Yıl : 2022

Sayfa Sayısı : 84

Anahtar Kelimeler : Tam metin arama, veri tabanı, MSSQL Server, MySQL, MongoDB, Elasticsearch, İVTYS, NoSQL.

Master's Thesis
Performance Comparison of Text Search Methods Used in Databases
Trakya University Institute of Natural Sciences
Department of Computer Engineering

ABSTRACT

In this thesis, Full Text Search (FTS: Full Text Search) methods in Relational and NoSQL databases, which are usually performed with special indexes called text indexes, are examined and performance comparisons are made. In text indexes, words or terms are matched to the documents in which they are located, and if necessary, information about how many times and in which locations the words are located in the documents can also be stored. While searching a word or term in a database, the search process is performed much faster using text indexes instead of scaning all documents. In recent years, many database management systems have started to offer FTS support, and search engines such as Elasticsearch, which are especially used for this purpose, have also emerged.

In the performance comparison section of the thesis, a large number of small-sized text data such as article abstracts and large-sized text data such as books have been added to MSSQL Server, MySQL, MongoDB and Elasticsearch databases. After the text indexes were created with the relevant methods, the words determined for the test were searched on these indexes and the time results were obtained. The same words were also searched with Regex/Like type text search queries and the results were compared with the results obtained using the index.

As a result of the study, it has been revealed that which Full Text Search or Regex/Like method works better on which database management system.

Year : 2022
Number of Pages : 84
Keywords : Full text search, database, MSSQL Server, MySQL, MongoDB, Elasticsearch, RDBMS, NoSQL.

TEŞEKKÜR

Tez çalışmam kapsamında, konu belirlemesinden savunmaya kadar gerek donanımsal gerek yazılımsal desteklerinin yanında, gece gündüz demeden yol gösteren ve çalışmamla ilgili danıştığım her probleme çözümler sunarak benden yardımlarını hiç esirgemeyen saygıdeğer tez danışmanım Dr. Öğr. Üyesi Altan MESUT hocama ve çalışmamla ilgili kendisine her danıştığımda yol gösteren Dr. Öğr. Üyesi Emir ÖZTÜRK hocama teşekkürlerimi sunarım.

Yüksek Lisans eğitimim boyunca kendilerinden ders aldığım hocalarım başta olmak üzere bana katkısı olan üniversitenin tüm personellerine de teşekkür ederim.

Hayatımın her aşamasında yanımda olan, sevgi ve desteklerini benden hiç esirgemeyen, haklarını hiçbir zaman ödeyemeceğim sevgili annem Fatime ABAKAR ve sevgili babam BRAHIM ADOUM'a sonsuz saygı, sevgi ve şükranlarımı sunarım.

Mahamoud BRAHIM ADOUM

İÇİNDEKİLER

ÖZET ..... **iv**
ABSTRACT ..... **v**
TEŞEKKÜR ..... **vi**
İÇİNDEKİLER ..... **vii**
SİMGELER DİZİNİ ..... **x**
ÇİZELGELER DİZİNİ ..... **xi**
ŞEKİLLER DİZİNİ ..... **xii**
BÖLÜM 1: GİRİŞ **1**
BÖLÜM 2: GENEL BİLGİLER **4**

2.1. Tam Metin Arama (Full Text Search, FTS) İşlemleri *4*
2.2. Tam Metin Aramada Dile Özgü İşlemler *4*
2.3. Ters Dizin (Inverted Index) *5*
2.4. Tam Ters Dizin (Full Inverted Index) *6*
2.5. İlişkisel Veri Tabanı (İVT) *8*
2.6. İlişkisel Olmayan Veri Tabanı (NoSQL) *9*
2.7. ACID ve BASE Modelleri *10*
2.7.1. ACID modeli 10
2.7.2. BASE modeli 10
2.8. Ölçeklendirme *12*
2.10. Tez Kapsamında Karşılaştırılan RDBMS Veri Tabanları *13*
2.10.1. MySQL 13
2.10.2. Microsoft SQL Server 14
2.11. Tez Kapsamında Karşılaştırılan NoSQL Veri Tabanları *14*
2.11.1. MongoDB belge deposu (Document Store) 14
2.11.1.1. Mimarisi ve kullanım alanları 17
2.11.1.2. Avantajları 18

2.11.1.3. Dezavantajları 19
2.11.2. Elasticsearch arama motoru (Search Engine) 19
2.11.2.1. Mimarisi ve kullanım alanları 20
2.11.2.2. Avantajları 22

BÖLÜM 3: MATERYAL VE YÖNTEMLER 24

3.1. MySQL'de Tam Metin Arama 25
3.1.1. Tam metin indeksi oluşturma 25
3.1.2. Oluşturulan indeks üzerinde arama yapma 26
3.1.3. FTS yapılandırması 28

3.2. Microsoft SQL Server'da Tam Metin Arama 29
3.2.1. FTS özelliğinin veri tabanında kullanılabilir hale getirilmesi 29
3.2.2. Tam metin kataloğu oluşturma 30
3.2.3. Tam metin indeksi oluşturma 30
3.2.4. Oluşturulan indeks üzerinde arama yapma 31

3.3. MongoDB'de Tam Metin Arama 32
3.3.1. Metin indeksi oluşturma 32
3.3.2. Oluşturulan indeks üzerinde arama yapma 33

3.4. Elasticsearch'te Tam Metin Arama 34

BÖLÜM 4: TEZ KAPSAMINDA SEÇİLEN VERİ TABANI YÖNETİM SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI 35

4.1. Kullanılan Cihazın Özellikleri 35

4.2. Test için Kullanılan Veri Setleri 35

4.3. Testlerde Kullanılan Veri Tabanı Sistemleri 36
4.3.1. MSSQL'de yapılan test öncesi işlemler 37
4.3.1.1. Veri tabanı oluşturma 37
4.3.1.2. Veri seti yükleme 37
4.3.1.3. İndeks oluşturma 38
4.3.2. MySQL'de yapılan test öncesi işlemler 39
4.3.2.1. Veri tabanı oluşturma 39
4.3.2.2. Veri seti yükleme 39
4.3.2.3. İndeks oluşturma 40
4.3.3. MongoDB'de yapılan test öncesi işlemler 40
4.3.3.1. Veri tabanı oluşturma 40
4.3.3.2. Veri seti yükleme 41
4.3.3.3. İndeks oluşturma 41
4.3.4. Elasticsearch'de indeks oluşturma ve veri seti yükleme 42
4.3.5. Veri ve indeks boyutlarının karşılaştırılması 43

4.4. Yapılan Sorgu Testlerinin Sonuçları *45*
4.4.1. İndeksler üzerinde yapılan FTS arama sonuçları 47
4.4.2. İndeksli olmayan veriler üzerinde yapılan Like/Regex arama sonuçları 52
BÖLÜM 5: SONUÇLAR **54**
KAYNAKLAR **56**
EKLER **60**
ÖZGEÇMİŞ **71**
TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER **72**

SİMGELER DİZİNİ

Kısaltmalar

ACID	Atomicity, Consistency, Isolation, Durability
BASE	Basically Available, Soft-state, Eventualy-consistent
BSON	Binary JSON
CAP	Consistency, Availability, Partition Tolerance
CRUD	Create, Read, Update, Delete
DML	Data Manipulation Language
DSL	Domain Specific Language
FK	Foreign Key
FTS	Full-Text Search
HTTP	Hyper Text Transfer Protocol
İVTYS	İlişkisel Veri Tabanı Yönetim Sistemleri
JSON	Javascript Object Notation
NoSQL	Not Only SQL
PK	Primary Key
RDBMS	Relational Database Management Systems
SSMS	SQL Server Management Studio
sn	Saniye
SQL	Structured Query Language
VT	Veri Tabanı
XML	Extensible Markup Language

ÇİZELGELER DİZİNİ

Çizelge 2.1. Tam ters dizin örneği 7

Çizelge 2.2. RDBMS ve MongoDB yapılarının karşılaştırılması 16

Çizelge 2.3. RDBMS ve Elasticsearch yapılarının karşılaştırılması 20

Çizelge 4.1. MSSQL Server'da oluşturulan FTS indekslerinin detaylı bilgileri 39

Çizelge 4.2. MySQL'de oluşturulan FTS indekslerinin detaylı bilgileri 40

Çizelge 4.3. MongoDB'de oluşturulan FTS indekslerinin detaylı bilgileri 42

Çizelge 4.4. Oluşturulan indekslerde süre karşılaştırılması 44

Çizelge 4.5. FTS aramalarında kullanılan kelimeler 45

Çizelge 4.6. Like/Regex aramalarında kullanılan kelimeler 45

Çizelge 4.7. FTS arama sonuçları (sn) 51

Çizelge 4.8. Like/Regex arama sonuçları (sn) 52

Çizelge 5.1. Arama türüne göre en iyi veri tabanları 55

ŞEKİLLER DİZİNİ

Şekil 2.1. Tam metin aramada dile özgü işlemler örneği 5

Şekil 2.2. Ters dizin oluşturma örneği 6

Şekil 2.3. İlişkisel veri tabanında saklanan iki tablo 8

Şekil 2.4. Xml türünde depolanan bir örnek 15

Şekil 2.5. JSON türünde depolanan bir örnek 16

Şekil 2.6. MongoDB'de BSON türünde depolanan bir örnek 17

Şekil 2.7. MongoDB veri tabanının çalışma mantığı 18

Şekil 2.8. Elasticsearch cluster örneği 21

Şekil 4.1 Saklanan verilerin ve metin indekslerinin boyutlarının karşılaştırılması 44

BÖLÜM 1

GİRİŞ

Dünya nüfusunun yaklaşık üçte ikisinin internet kullandığı günümüzde veri üretimi gittikçe artmaktadır. İnternet ortamındaki veriler genellikle yapılandırılmamış biçimdedir. Bu verilerin depolanması ve işlenmesi üretiminden daha fazla önem taşımaktadır. Yapısal biçiminde olan verilerin depolanmasında ilişkisel (Codd, 1970), yapısal olmayan verilerin depolanmasında ise NoSQL modelini kullanan veri tabanları tercih edilmektedir. NoSQL veri tabanlarını ortaya çıkaran nedenlerden biri de, ilişkisel veri tabanlarında ölçeklendirmenin dikey olarak yapılmasından dolayı karşılaşılan zorluk ve yüksek maliyettir (Dumanlı, 2019, s: 53).

Günümüzde, kurumsal ve özel şirketler başta olmakta üzere, sosyal medya platformlarında da veri tabanlarına saniyeler içinde terabaytlar boyutunda veri girişi gerçekleşmektedir. Aranan bir terim veya kelimenin böyle bir büyük veri (big data) içinde hangi dokümanlarda olduğunu bulup kullanıcıya göstermenin süresi ise kullanılan donanımın gücü kadar, verinin kaydedilme biçimine ve kullanılan veri tabanı yönetim sistemine bağlıdır.

Veri tabanlarında metin arama hızını arttırmak amacıyla metin indekslerinin kullanımı gittikçe yaygınlaşmaktadır. Metin indeksleri oluşturulurken, belgelerin yazıldıkları dile göre bazı ön işlemlerden geçirilerek metin indekslerindeki eleman sayısının azaltılması oldukça önemlidir. Bir kelimenin sadece kökü indekslenerek, farklı ekler almış hallerinin farklı indeks anahtarı olarak saklanması engellenebilir. Örneğin İngilizce diline göre bir ön işlem yapılarak ‘compute’, ‘computer’, ‘computation’ veya ‘computing’ kelimeleri ‘comput’ anahtarı ile indekslenerek, hepsinin arama sonuçlarının

aynı olması sağlanabilir. Bu şekilde hem indeks boyutu küçültülür, hem de indeks üzerinde arama hızı arttırılabilir.

Tam Metin Arama (Full-Text Search: FTS) için genellikle Ters Dizin (Inverted Index) yöntemi kullanılır. Ters dizin adından da anlaşıldığı gibi normal arama mantığı yönteminin yaptığı işin tersini yaparak aranan sonuca ulaşmaktadır. Normal arama yönteminde aranan doküman bulunup içindeki kelimeye ulaşılırken, ters dizin yönteminde özel indeksler temel alınarak hangi kelimenin hangi dokümanda yer aldığı bulunmaktadır (Ayan, 2017). Veri tabanlarında sıkça kullanılan B-Tree indeksleme yöntemi de, metin indekslerinde kullanılabilir. MongoDB'de klasik metin indekslemede B-Tree türünü kullanırken bulut veri tabanı hizmeti olan MongoDB Atlas içinde bulunan metin arama özelliği, Elasticsearch gibi Apache Lucene kütüphanesini kullanır, yani ters dizin ile çalışır.

Literatürde tam metin arama yaklaşımlarının performanslarını inceleyen bazı çalışmalar olsa da, hangi veri tabanında kullanılan metin arama yönteminin daha etkin çalıştığı konusunda bugüne kadar kapsamlı bir araştırma yapılmadığı görülmektedir. Kılıç ve Karabey (2016) tarafından yapılan çalışmada, popüler tam metin arama motorlarından Solr ve Elasticsearch verimlilik, kullanım kolaylığı, hız ve güvenlik açısından karşılaştırılmış ve her iki arama motorunun da avantaj ve dezavantajlarına yer verilmiştir. Minack, Siberski ve Nejdl (2009) tarafından yapılan çalışmada, sadece RDF depoları üzerinde tam metin arama performansı ölçülmüştür. Mesut ve Öztürk (2021) tarafından yapılan çalışmada ise MongoDB'nin BTree indeksi ile yapılan klasik tam metin arama işleminin daha hızlı çalışabilmesi için bir yaklaşım sunulmuştur.

Ülkemizde ve dünyada bazı veritabanları üzerinde CRUD (Create, Read, Update, Delete / Oluşturma, Okuma, Güncelleme, Silme) performansını ölçen ve kıyaslayan bazı çalışmalar yapılmış olsa da, bu çalışmalara FTS türü sorgulamaların performansı dahil edilmemiştir. Sadece NoSQL türündeki veri tabanlarının performans karşılaştırması üzerine bazı tezler yapılmış (Aladily, 2015; Bhamra,2017; Hammood, 2016; Shwaysh, 2018; Singh, 2015; Swaminathan, 2015), bazı tez çalışmalarında ise hem ilişkisel hem de NoSQL türünde olan veri tabanları beraber performans yönünden karşılaştırılmıştır (Bolat, 2019; Dumanlı, 2019; Seçgin, 2018; Taha, 2017).

Bu tez çalışmasının asıl amacı veri tabanları veya metin arama motorları üzerinde yapılan tam metin arama yöntemlerini incelemek, negatif ve pozitif yönlerini belirlemek ve performans karşılaştırması yapmaktır. Tez çalışmasının yan amaçlarından biri B-Tree ve ters dizin indeksleme yöntemleri arasındaki performans farkını ortaya koymak, diğeri de seçilen veri tabanı yönetim sistemleri üzerinde metin indeksi kullanmadan Like/Regex türü metin arama sorgularının da hızını karşılaştırmaktır. Bu tez çalışmasının özgün değeri, literatürde örneğine rastlanmayan veri tabanı yönetim sistemleri üzerindeki FTS işlemlerinin perforansının ortaya konulmasıdır. Bu amaca yönelik olarak MSSQL Server ve MySQL İlişkisel Veri Tabanı Yönetim Sistemleri (İVTYS - RDBMS: Relational Database Management Systems) ile MongoDB ve Elasticsearch NoSQL (Not Only SQL) veri tabanı yönetim sistemleri üzerinde testler yapılmıştır.

BÖLÜM 2

GENEL BİLGİLER

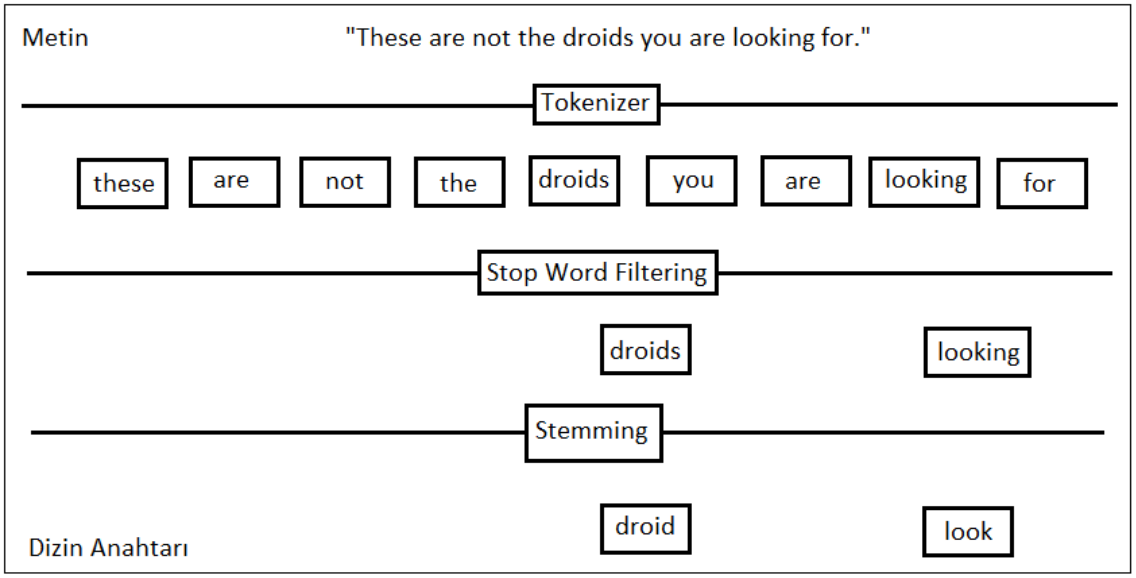
2.1. Tam Metin Arama (Full Text Search, FTS) İşlemleri

Tam Metin Araması kitap, makale, bildiri veya dergiler gibi elektronik biçiminde dokümanlar şeklinde belirli bir yerde tutulan kaynakların üzerinde kelime veya metin aramasına denir. Aranan bir kelime veya metnin hangi kayıt veya dokümanlarda yer aldığını belirlemek için indeksli veya indeksli olmayan bir metin arama yöntemi kullanılabilir. İndeks kullanılmadığında, bir Veri Tabanında (VT) bir kelime veya metnin bulunduğu kaydı/dokümanı bulmak istediğimizde, tüm veriler üzerinden tarama (scan) işlemi yapılması gerekmektedir. Metin araması için oluşturulan özel indeksler üzerinden arama yapıldığında ise sonuçların çok daha hızlı getirilmesi sağlanabilir. Dokümanlar yazıldıkları dile özgü bazı ön işlemlerden geçirilerek metin indeksindeki eleman sayısının daha az olması sağlanabilir.

2.2. Tam Metin Aramada Dile Özgü İşlemler

Tam Metin Aramada, dile özgü olarak bazı ön işlemler yapılmaktadır. Bu işlemler metni parçalara ayırma (Tokenizer), etkisiz kelimlerin çıkarılması (Stopword Filtering) ve kök-ek ayırma (Stemming) işlemleridir. Bu çalışmada İngilizce kitap ve makale özetleri test amaçlı kullanıldığı için veri tabanlarında indeksleme yapılırken İngilizce'ye göre bu ön işlemler uygulanmıştır. Bu nedenle bu bölümdeki şekillerde İngilizce örnekler kullanılmaktadır.

- **Tokenizing:** Metni parçalara ayırma işlemidir. Genellikle metin boşluk karakterlerinden parçalara ayrılır ve her parça (token) metnin içindeki bir kelimeye denk gelir.
- **Stopword Filtering:** İngilizcede yer alan “the”, “and” veya “that” gibi tek başına aranmayacak etkisiz kelimeleri indekse dâhil etmenin pek bir anlamı yoktur. Stopword Filtering aşamasında bu türden etkisiz kelimeler elenir.
- **Stemming:** Kelimelerin eklerine bakmaksızın kök kısımlarının indekslenmesini sağlayan işlemdir. Örneğin bir metin içerisinde waiting, waits ve waited kelimesi için sadece ortak kökleri olan wait kelimesi indekslenir. Böyle bir indeksleme yönteminde daha az indeks anahtarı oluştuğundan dolayı indeks boyu küçülür ve arama hızı artmaktadır. Aynı zamanda aranan bir kelimenin farklı ekler almış halini içeren dokümanlara da ulaşılmış olur. Tam metin aramada dile özgü yapılan ön işlemler Şekil 2.1’de (Trelle, 2013) detaylı bir şekilde verilmiştir.

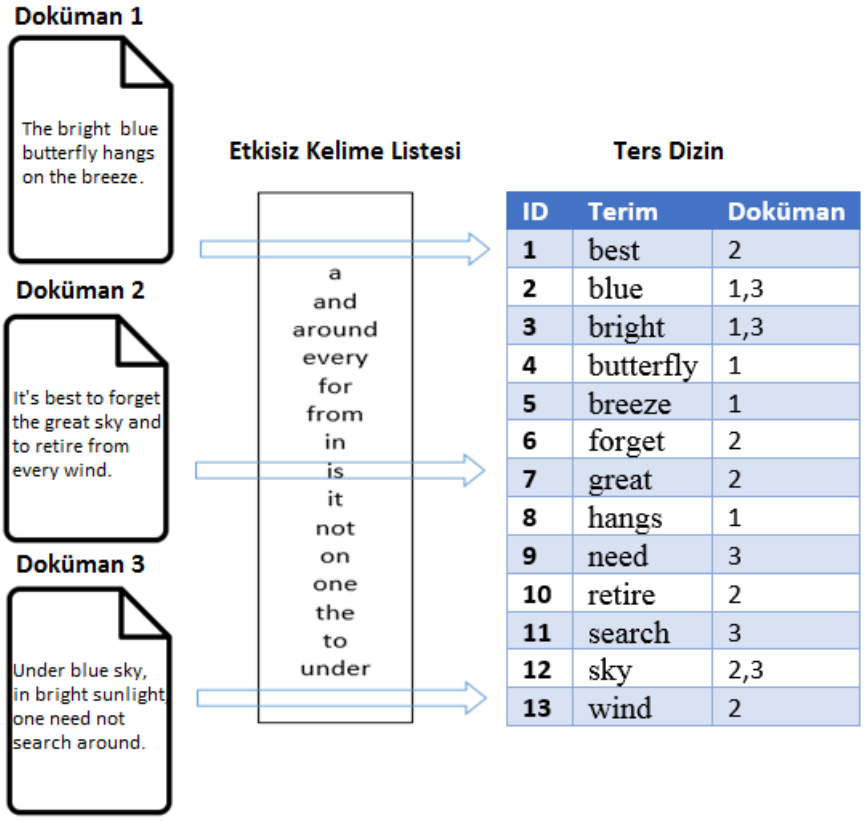


Şekil 2.1. Tam metin aramada dile özgü işlemler örneği.

2.3. Ters Dizin (Inverted Index)

Bu tür indekslemede, kelimelerin hangi dokümanlarda geçtiği saklanmaktadır. Normal arama işleminde dokümanlar üzerinden içerdikleri kelimelere ulaşılırken, ters dizin ise kelimeler üzerinden yer aldıkları dokümanlara ulaşmayı hedefler.

Bir veri seti üzerinde ters dizin oluşturulurken, tüm dokümanlarda yer alan kelimeler (önceki bölümde anlatılan ön işlemler sonrası geriye kalan kökler) ve bunların hangi dokümanlarda yer aldığı ayrı ayrı indekslenir. Ters dizinin oluşturulmasında yapılan işlem aşamaları Şekil 2.2'de (MentorMate, 2017) verilmiştir.



Şekil 2.2. Ters dizin oluşturma örneği.

Şekil 2.2'de görüldüğü gibi her kelimenin yanında yer aldığı dokümanlara karşılık gelen anahtarlar saklandığı için, bir kelimeyi içeren dokümanlara hızlı bir şekilde ulaşılabilir.

2.4. Tam Ters Dizin (Full Inverted Index)

Aranan terimlerin hangi dokümanlarda geçtiğinin yanı sıra terimin o dokümandaki kaçıncı kelime olduğu gibi bilgiler de saklanıyorsa, bu indeks türüne Full Inverted Index adı verilir. Şekil 2.2'de Ters Dizin olarak verilen örneğin üzerinde Tam Ters Dizin uygulayarak Çizelge 2.1'de verilmiştir.

Çizelge 2.1. Tam ters dizin örneği.

ID	Terim	Doküman
1	best	2 (2)
2	blue	1 (3), 3 (2)
3	bright	1 (2), 3 (5)
4	butterfly	1 (4)
5	breeze	1 (8)
6	forget	2 (4)
7	great	2 (6)
8	hangs	1 (5)
9	need	3 (8)
10	retire	2 (10)
11	search	3 (10)
12	sky	2 (7), 3 (3)
13	wind	2 (13)

Dizinde saklanan kelimelerin sıra numaraları sayesinde Çizelge 2.1'de verilen dizinde "bright blue" gibi iki kelimelik bir ifadeyi içeren dokümanlar da aranabilir. Doküman 1'de 'bright' 2 ve 'blue' 3 numaralı kelimeler olduğu için bu ardışık durum aramamız ile uyuşacaktır. Doküman 3'te de her iki kelime de yer almasına rağmen birbirlerine uzak konumda oldukları için arama ile uyuşmayacaktır.

Aynı kelime bir dokümanın onlarca veya yüzlerce farklı konumunda yer alıyor olabilir. Böyle durumlarda tüm konum bilgilerinin saklanması gerekeceğinden bu türde tam ters dizinler oldukça fazla yer kaplayacaktır.

Bazı indekslerde sadece aranan kelimelerin o dokümanlarda kaç defa geçtiğinin bilgileri saklanır. Böyle indeks türlerinde konum bilgisi saklanmadığı için "bright blue" gibi çok kelime içeren ifadelerin aranmasına izin verilmez. Fakat aranan kelimeyi daha fazla sayıda içeren dokümanların sonuç listesinde daha üst sıralarda gösterilmesine yardımcı olur.

2.5. İlişkisel Veri Tabanı (İVT)

Bir ilişkisel veri tabanı, birbiriyle ilişkili verilerin anahtar adı verilen sütunlar ile bağlantı kurularak depolanmasını sağlayan ve bunlara sonradan erişilmesine imkân tanıyan bir veri tabanı türüdür (Oracle Türkiye, 2021).

İlişkisel veri tabanlarında Satır (Row) ve Sütunlardan (Column) oluşan Tablo (Table) adı verilen yapılar mevcuttur. Satırlarda saklanan her kayıt içinde sütunlarda tanımlanan alanlara ait veriler yer alır. Tabloda birincil anahtar (primary key: PK) olarak tanımlanan alan tüm kayıtlarda farklı (tekil: unique) değerlere sahip olur. Birçok ilişkisel veri tabanında birincil anahtar olan alan otomatik olarak indekslenir. Bir tablonun birincil anahtarı ile farklı bir tablonun bir alanı ilişkilendirilirse, diğer tablodaki ilgili alana dış anahtar (foreign key: FK) adı verilir. Dış anahtar olan alana girilen verilerin birincil anahtar tarafında mutlaka bulunması gerekir. Dış anahtar tekil özelliğine sahip olmak zorunda değildir. Örneğin Şekil 2.3'te Bölümler ve Personel adında iki farklı tablo gösterilmektedir. Altı çizili olan PerNo ve BölümNo alanları birincil anahtar olan alanlardır. Bölümler tablosunda PK olan BölümNo alanı ile Personel tablosunun Bölümü alanı arasında bir ilişki kurulursa, Bölümü alanı FK olacaktır. Bir bölümde çalışan birçok personel olabileceği için bu alanda tekrar eden veriler olabilecektir (örnekte 10 değerinin tekrar etmesi gibi).

PERSONEL				
PerNo	**Adı**	**Soyadı**	**Görevi**	**Bölümü**
1023	Selim	Korkmaz	Memur	10
3925	Fatih	Şahin	Şef	10
1233	Esra	Yılmaz	Müdür	20

BÖLÜMLER	
BölümNo	**BölümAdı**
10	Pazarlama
20	Satış
30	Muhasebe

Şekil 2.3. İlişkisel veri tabanında saklanan iki tablo.

İki tablo arasında bir ilişki kurulursa, bu ilişkiye ait bazı kurallar tanımlanabilir. Bu kurallar sayesinde PK tarafı güncellendiğinde FK tarafındaki ilgili kayıtların da yeni değer ile güncellenmesi, PK tarafından bir kayıt silinirse FK tarafındaki ilgili kayıtların da silinmesi gibi işlemler veri tabanı yönetim sistemi tarafından otomatik olarak yapılabilir.

Veri tabanı yönetim sistemlerinde tablolar dışında indeks (index), görünüm (view), kullanıcı (user) ve rol (role) gibi farklı nesneler, prosedür (procedure), fonksiyon

(function) ve tetikleyici (trigger) gibi VT programlama dili elemanlarını oluşturmak da mümkündür.

2.6. İlişkisel Olmayan Veri Tabanı (NoSQL)

Son yıllarda verilerin inanılmaz boyutlara ulaşması ilişkisel veri tabanlarının sınırlarını zorlamaktadır. Bazı sosyal medya, mobil ve web platformlarında milyarlarca veri milisaniyeler içinde kaydedilmekte, aynı şekilde milyarlarca kullanıcı tarafından milisaniyeler içinde verilere eşzamanlı olarak erişme istekleri yapılabilmektedir. Böyle bir hızla çoğalmakta olan verileri ilişkisel veri tabanı ile anlık olarak toplayıp işlemek işlerin yavaş ilerlemesine yol açabilir. Bu gibi durumlarda NoSQL türünde veri tabanı yönetim sistemlerine başvurmak oldukça mantıklıdır. Günümüzde, Büyük Veri projelerinde, sosyal medya devlerinin, alışveriş sitelerinin, büyük iş merkezlerinin ve devletlere ait devasa kurumlarının verileri büyük hızda toplamaları, incelemeleri ve işlemeleri çoğunlukla NoSQL destekli veri tabanlarına borçlu olduğunu söyleyebiliriz.

NoSQL terimi "SQL'e hayır" gibi algılansa da birçoğu sorgu dili olarak SQL kullanır. Bu nedenle NoSQL bazı kaynaklarda "Not Only SQL – Sadece SQL Değil" ("NoSQL", 2022) olarak da tanımlanabilmektedir. NoSQL VT sistemleri, Key-Value Store (Anahtar – Değer Depo), Column-Oriented Store (Sütun Tabanlı Depo), Document-Oriented Store (Belge Tabanlı Depo), Graph database gibi farklı türlerde olabilmektedir.

İlişkisel veri tabanlarında tablolarda saklanan kayıtların içerdiği alanların özellikleri (alanın ismi, veri türü, boyutu, vb.) tablonun yaratılması sırasında verilmektedir. Tabloda saklanan her kayıt bu yapıya (şemaya) uygun olmak zorundadır. NoSQL veri tabanlarının çoğunda ise belirli bir kayıt yapısı tanımlanmadığı için bu VT'ler yapısal olmayan (unstructured) verilerin saklanmasında elverişlidirler. Bu durum "şema bağımsızlığı" olarak bilinmektedir.

NoSQL veri tabanlarının ekleme, silme ve güncelleme işlemlerinde oldukça hızlı olması BASE modelinden kaynaklanmaktadır. İlişkisel veri tabanlarındaki ACID garantisi verilerin tutarlılığı için çok önemli olsa da veri ekleme veya güncelleme hızına olumsuz etkilemektedir. ACID yerine BASE yöntemini temel alan ve birçoğu yatay ölçeklendirme yapısına sahip NoSQL veri tabanları büyük verilerin saklanması ve işletilmesi için daha mantıklı bir çözümdür.

2.7. ACID ve BASE Modelleri

2.7.1. ACID modeli

İlişkisel veri tabanı sistemlerinde yapılan bir işlem (transaction) esnasında, herhangi bir veri kaybı yaşanmadan gerçekleşmesi ACID (Atomicity, Consistency, Isolation and Durability – Atomiklik, Tutarlılık, İzolasyon ve Dayanıklılık) tarafından garanti edilmektedir. Bu kavramlar aşağıda açıklanmıştır:

- **Atomiklik:** Bir işlemin ya bütünüyle gerçekleştirilen ya da hiç yapılmayan bölünmez bir birim olmasını sağlayan "ya hep ya hiç" özelliğidir. Bir işlem yürütülürken veri tabanına erişim problemi, elektrik kesintisi veya ağ problemi gibi sorunlar oluşursa yapılan işlem geçersiz kılınarak veri bütünlüğünün korunmasını sağlanır.
- **Tutarlılık:** Veri tabanında saklanan birbiri ile ilişkili verilerin belirli kurallar ve kısıtlamalar ile tutarlılı olmasını sağlayan özelliktir. Tutarlılığın veri tabanı tarafından sağlanması için ilgili tablolar üzerinde çeşitli bütünlük kısıtlamaları (integrity constraints) ve tetikleyiciler (triggers) tanımlanabilir.
- **İzolasyon:** Eş zamanlı olarak aynı veri üzerinde birden fazla işlem yapılması istenildiğinde, işlemlerin birbirinden bağımsız olarak yürütülmesini sağlayan özelliktir. Bir işlem tamamlanana kadar (commit edilene kadar), değiştirdiği verilerin diğer işlemler tarafından görülmemesi (izole olması) sağlanır.
- **Dayanıklılık:** Başarıyla tamamlanan işlemleri veri tabanına kaydetmeyi, fiziksel veya işlemsel arıza meydana gelme durumunda ise sistemin kendisini bir önceki geçerli veri durumuna götürmesini sağlayan özelliktir.

Genellikle RDBMS sistemlerinde kullanılan ve verinin tutarlı olmasını ön planda tutan ACID modelinde kilit mekanizmaları çok fazla devreye girdiği için, özellikle DML (Data Manipulation Language - Veri İşleme Dili) kullanılarak yapılan ekleme-silme-güncelleme işlemlerinin yavaşlamasına neden olmaktadır.

2.7.2. BASE modeli

BASE (Basically Available, Softstate, Eventually-consistent / Basit Kullanılabilirlik, Esnek Durum, Nihai Tutarlılık), İlişkisel Olamayan Veri Tabanı

Sistemlerinde (Non-Relational Database Systems, NoSQL) yer alan BASE yöntemi RDBMS sistemindeki ACID kavramının zıttı olarak görülmektedir. BASE, esasen gerçek dünyada tam tutarlılığın elde edilemeyeceği ve bu nedenle yüksek oranda ölçeklenebilir dağıtılmış sistemler modellenemeyeceği gerçeğini benimsemektedir. BASE baş harflerinin temsil ettiği anlamlar aşağıdaki satırlarda detaylı bir şekilde anlatılmıştır (Gökşen & Aşan, 2015):

- **Basit Kullanılabilirlik:** Sistem CAP (Consistency, Availability, Partition Tolarance - Tutarlılık, Erişebilirlik, Bölünebilme Toleransı) teoremine uygun olarak çalışmaktadır. Herhangi bir hata durumunda bile sistem tüm taleplere cevap verir. Dolaysıyla verinin tutarlı olmasını ve verinin son haline erişildiğini garanti etmez. Başka bir deyişle verinin bir kısmını es geçerek daha basit bir erişilebilirlik sağlamaktadır.
- **Esnek Durum:** Verilerin tüm cihazlarda benzer şekilde görünmesinin garantisini vermez. Bu iş daha çok geliştiriciyi ilgilendirir. Kısacası, bu aşamada tutarlılık garantisi yoktur.
- **Nihai Tutarlılık:** Davranışına göre bu sisteme tutarlı denilebilir. Sistemin durumuna bağlı olarak İşlemlerin (Transaction) etkileri bir süre sonra diğer cihazlara yansımaktadır.

CAP teoremi ilk olarak Profesör Eric A. Brewer tarafından 2000 yılında gerçekleştirdiği bilişimle ilgili bir konuşmada ortaya atıldığı için Brewer Teoremi olarak da anılır (IBM Cloud Education, 14 Kasım 2019). Bu teorimin temelinde yatan mantık, dağıtık bir sistemde veri üzerinden sunulan hizmet için aynı anda üç özelliğinin sağlanamayacağıdır. Yani aynı anda ya CA (Consistency, Availability), ya CP (Consistency, Partition Tolarance) ya da AP (Availability, Partition Tolarance) gerçekleşebilir. CAP baş harflerinin temsil ettiği anlamlar aşağıdaki gibidir (Dayıbaşı, 2016):

- **Tutarlılık:** Hangi düğüme bağlanırlarsa bağlansınlar, tüm istemcilerin aynı verileri aynı anda görmesi anlamına gelir. Bunun olması için, bir düğüme veri yazıldığında, yazma işleminin 'başarılı' olarak kabul edilmesinden önce, sistemdeki diğer tüm düğümlere anında iletilmesi veya çoğaltılması gerekir.

- **Kullanılabilirlik:** Bir veya daha fazla düğüm kapalı olsa bile, veri talebinde bulunan herhangi bir istemcinin yanıt alması anlamına gelir. Dağıtık sistemdeki tüm çalışan düğümler, istisnasız herhangi bir istek için geçerli bir yanıt döndürür. Bu yanıt en son yazılan veri olmayabilir.
- **Bölüm Toleransı:** Dağıtık sistemdeki düğümler arasındaki herhangi bir sayıda iletişim kesintisine rağmen kümenin çalışmaya devam etmesi gerektiği anlamına gelir.

Özet olarak ACID modeli RDBMS'lerde sıkça kullanılıyor olup veri tutarlılığına önem vermekteyken; BASE modeli ise verinin erişebilirliğini ön plana sürerek NoSQL sistemlerinde sıkça kullanılan bir modeldir. Fakat MongoDB, NewSQL ve Neo4J gibi bazı NoSQL sistemleri ACID modelini desteklemektedir.

2.8. Ölçeklendirme

Veri tabanlarının performansı, gücü ve zayıflığını ölçebilmek için çeşitli parametreler mevcuttur ve ölçeklenebilirlik bunlardan biridir. Ölçeklenebilirlik, sürekli olarak artmakta olan işleri kaldırabilmesi veya uyum sağlaması amacıyla bir sisteme kaynaklar ekleyerek işin üstesinden gelebilmesini sağlama işlemine denir. Veri tabanlarının Dikey ve Yatay olarak ölçeklendirilmesi mümkündür:

- **Dikey Ölçeklendirme (Vertical Scaling – Scale Up):** Tek bir cihazın üzerine CPU, RAM gibi daha fazla kaynağın eklenmesi anlamına gelmektedir. Günümüzde yazılım sektörünün gelişmesi sayesinde bu tür teknikler, sanallaştırma teknolojisiyle kolaylaşmıştır (Swaminathan, 2015).
- **Yatay Ölçeklendirme (Horizontal Scaling – Scale Out):** Daha ucuz birçok cihazı birleştirerek güçlü bir sistem oluşturma esasına dayanmaktadır. Böyle bir sistemi büyütüp küçültmek daha az maliyetli ve daha az riskli olmaktadır (Swaminathan, 2015).

Genellikle RDBMS sistemler dikey, NoSQL sistemler de yatay olarak ölçeklenmektedir.

2.9. Arama Motorları

Arama motorlarına en çok Web siteleri ihtiyaç duyuyor olsa da günümüzdeki neredeyse tüm platformlarda yer aldığını görmekteyiz. Arama motorlarının arka planında bir nevi tam metin arama işleri gerçekleşmektedir. Dünyaca ünlü olan Google arama motoru örnek açısından verilebilir. Çalışmamızdaki testlerde kullandığımız Elasticsearch arama motoru ve onunla aynı altyapıyı kullanan Apache Solr günümüzde yaygın olarak kullanılmakta olan Tam Metin Arama motorlarıdır.

Elasticsearch ve Solr'nin alt yapısında Apache Lucene kütüphanesi vardır. Lucene Java ile geliştirilmiştir ve .NET versiyonu ile de .NET kullanıcılarının da kullanımına sunulmuştur (Gökalp, 2015).

Apache Solr, açık kaynak kodlu bir platformdur. Arama motoru olmanın yanında verilerin replikasyonu, çok çekirdekli çalışma imkanı ve HTTP gibi özellikleriyle de öne çıkar. Solr aynı zamanda HTTP istekleriyle sorgular yapabileceğiniz kur-çalıştır mantığında kullanıma hazır bir web uygulamasıdır. Bu özelliğiyle Lucene'nin paketlenmiş hali denilebilir (Ayan, 2017).

2.10. Tez Kapsamında Karşılaştırılan RDBMS Veri Tabanları

2.10.1. MySQL

MySQL, dünyanın en çok kullanılan ikinci veri tabanı yönetim sistemi (Db-Engines, 2022) olmanın yanı sıra, açık kaynak kodlu ilişkisel veri tabanı yönetim sistemlerinin en popüleridir. Günümüzde Oracle Corporation tarafından geliştirilmekte, dağıtılmakta ve desteklenmektedir. Adını ürünün yaratıcılarından biri olan Michael Widenius'un My adındaki kızından almış ve ilk sürümü 1995 yılında piyasaya çıkmıştır ("MySQL", 2022).

MySQL'in gelişmiş özelliklerinden faydalanmak isteyen kullanıcılar belli bir ücret karşılığında özel sürümler edinebilirler. Bununla beraber sağlamış olduğu esneklik, ölçeklenebilirlik, güvenilirlik ve hızlılığı sayesinde ilişkisel veri tabanları arasında çok tercih edilmektedir. Veri tabanı gerektiren neredeyse tüm platformlarda kullanılıyor olmasının yanı sıra en çok Web projelerinde tercih edilmektedir. Web programlama

dillerinde PHP ile daha fazla kullanılmaktadır. Windows, macOS, Linux ve Solaris gibi birçok işletim sistemine kurulabilmektedir.

MySQL sisteminde veri tabanlarını yaratma ve yönetme işlemleri için PHP dili ile geliştirilmiş web tabanlı PhpMyAdmin yazılımı sıkılıkla kullanılmaktadır. MySQL verilerini ayrıca MySQL Workbench yazılımı ile yönetmek de mümkündür.

2.10.2. Microsoft SQL Server

MSSQL Server ilişkisel veri tabanı yönetim sistemi, 1989 yılında Microsoft firmasının ilk veri tabanı olarak SQL Server v1.0 adı altında ve 16 bitlik olan OS/2 işletim sistemi üzerinde çalışacak şekilde piyasaya çıkarılmıştır. OS/2 işletim sistemi piyasadan silinince uzun yıllar sadece Microsoft Windows işletim sistemi altında çalışan SQL Server, 2017'den itibaren Linux işletim sisteminde de kullanılabilmektedir. Microsoft'un 1992'de piyasaya sürdüğü dosya temelli daha basit Access veri tabanından farklı olarak, SQL Server istemci/sunucu (client/server) türündedir.

Zaman içinde sürekli gelişen MSSQL Server, günümüzde en çok kullanılan üçüncü veri tabanıdır (Db-Engines, 2022). Java, C++, C#, ASP.NET gibi birçok yazılım geliştirme dilleriyle uyumlu çalışmaktadır. MSSQL Server, kaydedilen verilerin güvenliğini, bütünlüğünü ve aynı anda birden fazla kullanıcı tarafından erişilmesini garanti eder. Access'te olmayan kullanıcı yönetimi mekanizmasına sahiptir. Veri tabanı kullanıcılarının hangi verilere erişebilecekleri ve değiştirebilecekleri kapsamlı bir biçimde tanımlanabilir. Ayrıca verilerin sıkıştırılarak saklanması, hassas verilerin şifrelenmesi gibi ek özellikler de sunulmaktadır.

SQL Server Management Studio (SSMS) aracılığıyla MSSQL veri tabanında tablo, indeks, kullanıcı gibi tüm VT nesneleri yaratılabilir, prosedür, fonksiyon ve tetikleyici gibi programlama elemanları yaratılabilir ve tüm diğer veri tabanı yönetim işlemleri gerçekleştirilebilir.

2.11. Tez Kapsamında Karşılaştırılan NoSQL Veri Tabanları

2.11.1. MongoDB belge deposu (Document Store)

MongoDB, 2007 yılında daha sonra adını "Mongodb Inc" ile değiştiren 10gen şirketi tarafından C++ ile geliştirilmeye başlanmıştır (Mesut & Öztürk, 2021).

Günümüzde en çok tercih edilen açık kaynak kodlu ve belge tabanlı NoSQL veri tabanıdır (DB-Engines, 2022). İlk versiyonu 2009 yılında kullanıma sunulmuştur (“MongoDB”, 2022). Esnek, öğrenilmesi kolay ve oldukça hızlı bir veri tabanıdır.

Belge tabanlı veri tabanı sistemleri, Lotus Notes’tan esinlenmiş olup veriler bir anahtara karşılık gelen ve belge adı verilen nesnelerde saklanmaktadır. Bu sistemlerde belgeler genellikle JSON (Javascript Object Notation) veya XML (Extensible Markup Language) şeklinde saklanmaktadır. Anahtar - Değer NoSQL veri tabanlarından farklı olarak dokümanların içeriği bilinebilir ve sorgulanabilir. MongoDB, CouchbaseDB, CouchDB, Firebase Realtime Database, MarkLogic, RethinkDB ve Cloudant bu tür veri tabanlarına örnek olarak verilebilir. Aynı verilerin JSON ve XML biçiminde saklanmasıyla ilgili örnekler sırasıyla Şekil 2.4 ve Şekil 2.5’te verilmiştir:

```
<tezbankasi>
     <tezler>
          <tez>
               <id>23590</id>
               <konu>Veri Tabanı</konu>
          </tez>
          <tez>
               <id>90235</id>
               <konu>Büyük Veriler</konu>
          </tez>
     </tezler>
<tezbankasi>
```

Şekil 2.4. XML türünde depolanan bir örnek.

```
{
     "tezbankasi":
     {
          "tezler":
          [
               {
                    "id": 23590,
                    "konu": "Veri Tabanı",
               },
               {
                    "id": 90325,
                    "konu": "Büyük Veriler",
               }
          ]
     }
}
```

Şekil 2.5. JSON türünde depolanan bir örnek.

Çizelge 2.2'de MongoDB ve RDBMS veri tabanlarında kullanılan yapıların karşılaştırılması verilmiştir.

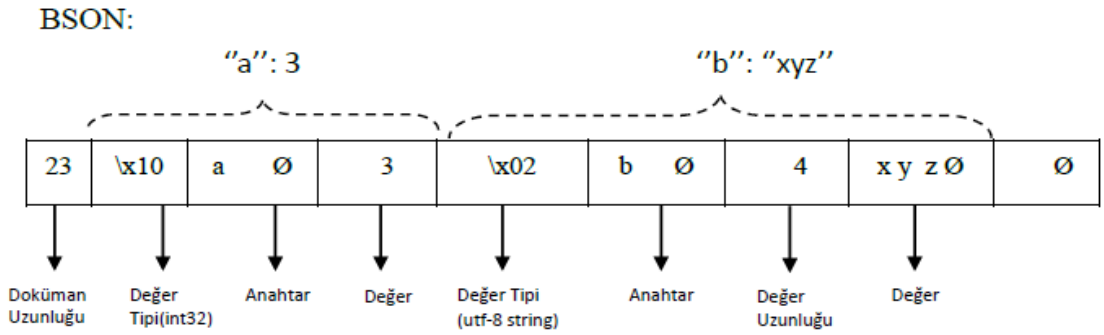
Çizelge 2.2. RDBMS ve MongoDB yapılarının karşılaştırılması.

RDBMS	**MongoDB**
Veri Tabanı	Veri Tabanı
Tablo	Koleksiyon
Satır	Doküman
Sütun	Alan
Tablo İlişkisi	Gömülü Dokümanlar

İlişkisel veri tabanından farklı olarak MongoDB'de, aynı koleksiyonda farklı sayıda ve farklı veri tiplerinde olan farklı alanlara sahip belgeler yer alabilir. Örneğin Personel bilgilerinin saklandığı bir koleksiyonda, Şekil 2.3'teki gibi önceden tanımlanmış belli alanlara sahip kayıtlar yerine, bazıları doğum tarihi, bazıları ise maaş bilgisini içeren farklı belgeler olabilir.

MongoDB veri tabanında veriler BSON (Binary JSON) diye adlandırılan JSON yapısının ikili karşılığı biçiminde depolanmaktadır. BSON, JSON formatındaki dokümanları bilgisayar ağı üzerinden etkin aktarımı için tasarlanmış bir formattır.

MongoDB veri tabanının benzersiz sorgulama dili sayesinde devasa veriler üzerinde çok kolay ve hızlı işlemler yapılabilmektedir. Windows, Linux, Solaris ve FreeBSD gibi birçok işletim sistemi üzerinde çok rahat bir şekilde kurulup çalıştırılabilmektedir. Python, Java, JavaScript, Go, C, C++, Ruby, Perl ve PHP gibi sıkça kullanılan hemen hemen tüm programlama dilleri ile uyumlu çalışmaktadır. BSON türünde saklanan bir verinin detaylı bilgileri Şekil 2.6'da (Mesut & Öztürk, 2021) verilmiştir.



Şekil 2.6. MongoDB'de BSON türünde depolanan bir örnek.

Compass ve Robot 3D gibi grafiksel arayüzler ile MongoDB'de veri tabanı ve koleksiyonlar yaratmak, CRUD işlemleri yapmak mümkündür.

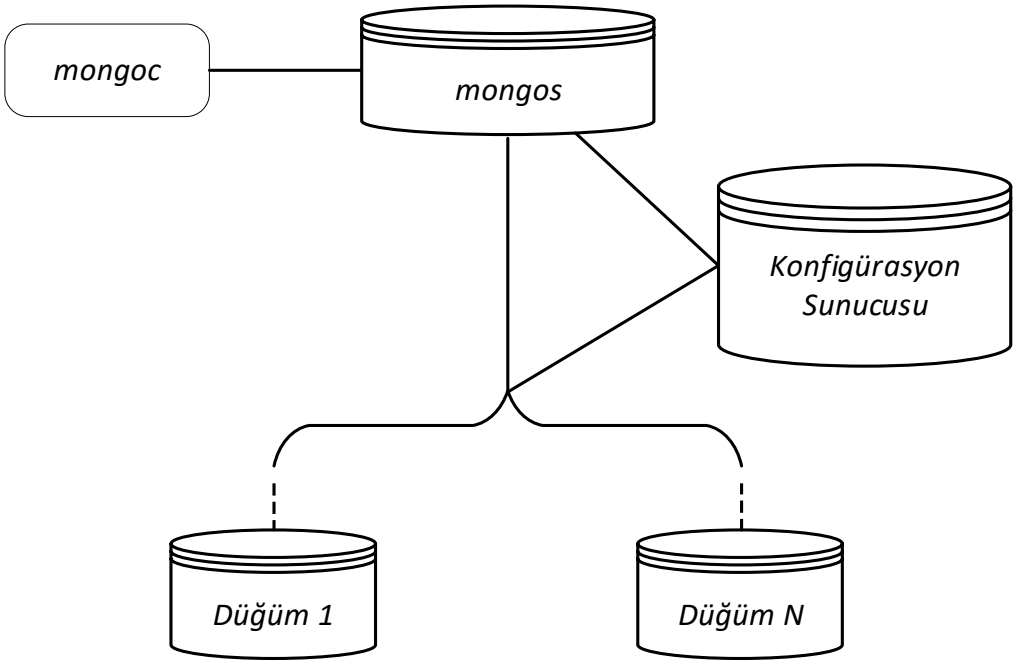
2.11.1.1. Mimarisi ve kullanım alanları

MongoDB, istemci-sunucu (client-server) mimarisine sahiptir. JSON türündeki dokümanlara hızlıca erişim sağlayabilmek, onları bellekte kalıcı hale getirebilmek için BSON türünde saklamaktadır. Sunucuyu başlatmak için hizmetler altında bulunan *mongod process* çalıştırılabilir.

MongoDB, zaman gittikçe sürekli yapısal güncelleme gerektiren uygulamalar, coğrafi veya günlük verileri depolamak, çok sunucu gerektirebilecek dağıtık uygulamalar ve büyük veri projeleri gibi alanlarda kullanılmaktadır (Bhamra, 2017). MongoDB mimarisinde aşağıdaki üç temel yapı yer almaktadır.

- **Shard (Düğüm):** Veri tabanına eklenen verinin belirli bir kısmını saklamakla yükümlüdür.

- **Config Server (Konfigürasyon Sunucusu):** Bu yapı, düğümlere bölünmüş sistemin üst bilgisini saklamaktadır. Bu üst bilgiler, tüm veri ve bileşenlerin bölünmüş sistem içerisindeki yapısını içerir. Ayrıca her düğümdeki yığınları (chunk) ve bu yığınların sınırlarını belirleyen aralıkları da içermektedir.
- **MongoDB Shard Server (mongos):** Düğümler üzerinde sorgu yürütmek ve veri yazma işlemlerinden sorumludur. Konfigürasyon Sunucularından aldığı üst bilgi aracılığıyla bu işlemleri gerçekleştirmektedir. MongoDB çalışma mantığı Şekil 2.7'de (Mesut & Öztürk, 2021) detaylı olarak verilmiştir.



Şekil 2.7. MongoDB veri tabanının çalışma mantığı.

2.11.1.2. Avantajları

- FTS desteğini bulundurduğu için üzerinde metin indeksleri oluşturulabilir ve bunların üzerinde aramalar yapılabilir.
- Çok esnektir ve anahtara bağlı olarak tüm değer setlerini barındırabilir.
- BASE modeli sayesinde veri eklemede yüksek performans gösterebilir.
- Ölçeklendirme özelliği sayesinde büyük verilerin sunucular arasında paylaştırılmasına imkân sağlar.

- Dokümanın her niteliğine ve anahtarına göre indeksleme yapılabilir. İndeks oluşturduktan sonra sorgularda hızlanma olurken, toplam (aggregate) fonksiyonlarında indeksin bir etkisi görülmemektedir.

2.11.1.3. Dezavantajları

- Veri yazma esnasında birleşme özelliği olmadığından dolayı her bir değer çifti için anahtar isimleri saklayarak veri tekrarına, dolayısıyla bellek ve disk alanlarında fazla yer kullanımına neden olur.
- Kötü bir indeksleme konfigürasyonu fazla parçadan oluşabilir ve düzensiz bir yapı üretebilir. Bu da sistemin çok yavaşlamasına neden olur.
- MongoDB disk alanını otomatik olarak temizleyemez. Örneğin dokümanların güncellenmesi veya silinmesi söz konusu olduğunda yeni bir disk alanı açamamaktadır. Dolaysıyla bu işlemin manüel olarak yapılması gerekmektedir.
- RDBMS sistemlerinde olan JOIN benzeri bir sorgulama tipi yoktur. Bu durum daha çok kod parçaları yazmaya neden olabilir.
- MongoDB'de tetikleyici (trigger) gibi bir yapı bulunmamaktadır.

2.11.2. Elasticsearch arama motoru (Search Engine)

Elasticsearch, Java dilinde yazılmış, açık kaynak kodlu, Apache Lucene altyapılı olan bir arama motorudur (Sumukh & Nandita, 2020). Aynı zamanda bir veri tabanı olup verileri dokümanlar şeklinde JSON olarak indekslemektedir. Dağıtık ve ölçeklenebilir şekilde çalışmaktadır (Balıkçı, 2020). Elasticsearch metin üzerinden doğrudan arama yapmak yerine, indeksler üzerinden arama yapar ve çok hızlı bir şekilde sonuçlar üretir. Bunun yanında sorgular üzerinde istatiksel analizler, skorlamalar da yapabilir (Doğan, 2019).

Elasticsearch, RestfulAPI üzerinden hizmet verdiğinden dolayı tüm programlama dilleriyle kolayca kullanılabilmektedir. Yüksek erişilebilirlik, veri tiplerine uygun formatta otomatik haritalama (mapping) yapabilmektedir. Büyük veriler üzerinde oldukça hızlı arama yapabilme kabiliyetine sahiptir (Sumukh & Nandita, 2020).

Çizelge 2.3'te Elasticsearch ve RDBMS veri tabanlarında kullanılan yapıların karşılaştırılması verilmiştir.

Çizelge 2.3. RDBMS ve Elasticsearch yapılarının karşılaştırılması.

RDBMS	Elasticsearch
Katalog veya Veri Tabanı	Küme
Tablo	İndeks
Satır	Doküman
Sütun	Alan
Şema	Örtük

Elasticsearch ortamına veri yükleme kendine has kütüphaneler veya RestfulAPI kullanılarak yapılmaktadır. Ayrıca River ve benzeri gibi veri yükleme araçları vasıtasıyla Wikipedia veya Twitter üzerinden edindiğimiz verileri yüklemek oldukça kolaydır (Balıkçı, 2020). Bunlarla beraber MongoDB ve diğer NoSQL veri tabanlarından veri aktarımı sağlanabilmektedir. Elasticsearch ara yüzü olan Kibana sayesinde, ".JSON", ".TXT", ".CSV" formatındaki çeşitli veriler Elasticsearch ortamına kaydedilebilmektedir. Kaydedilen veriler üzerinde dönüşümler gerçekleştirmek ve verileri çeşitli hedeflere aktarabilmek için Logstash aracı kullanılabilir.

Kibana, görsel grafikleri sayesinde büyük hacimli verilerin anlaşılmasını kolaylaştırmaktadır. Yerel bilgisayarlarda kullanımı sınırlıdır (Yang vd., 2019).

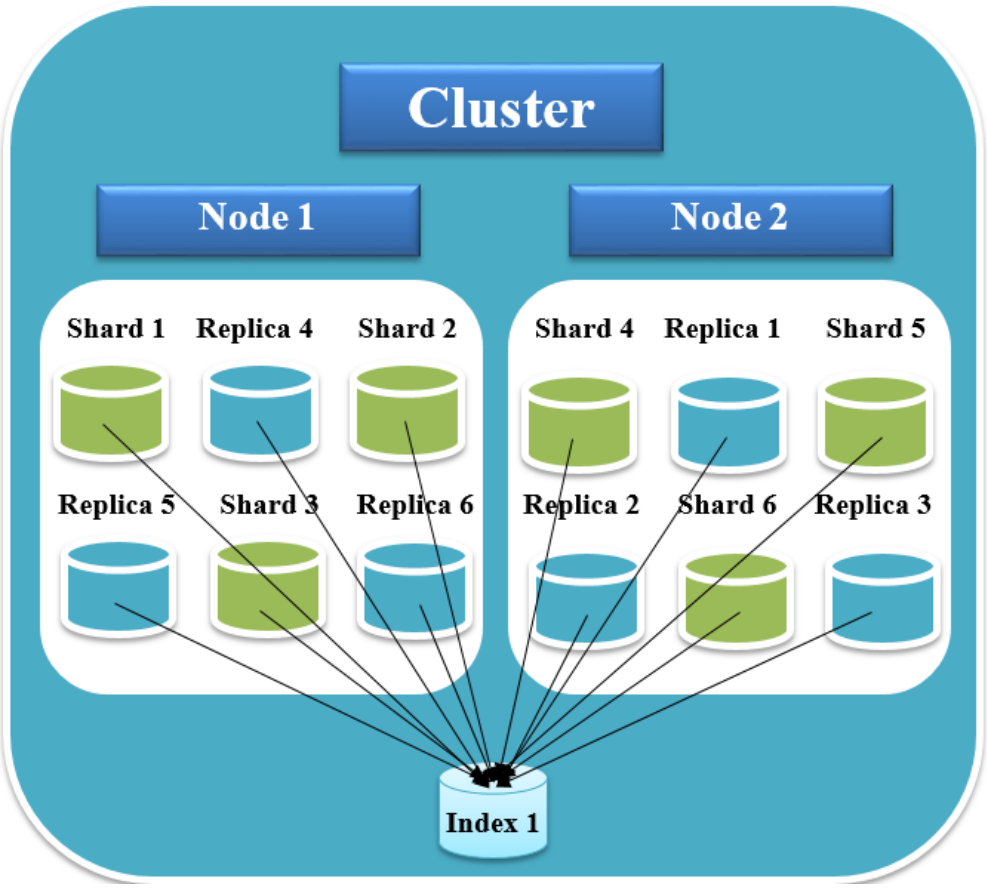
2.11.2.1. Mimarisi ve kullanım alanları

Elasticsearch mimarisine ait bazı önemli terminolojiler açıklamalarıyla beraber aşağıda verilmiştir (Delen, 2021):

- **İndeks:** Klasik arama motorlarında aramalar veri tabanları üzerinden gerçekleştirilirken, Elasticsearch'te "Index" olarak adlandırılan indeks dosyaları üzerinden gerçekleştirilmektedir. Elasticsearch Cluster'larında birden fazla veri tabanı (Index) bulunabilir.
- **Mapping:** İndekslenen verilerin hangi veri tipinde olduğunun tanımlandığı işlemdir. String, integer, double ve boolean bunlara olarak verilebilir.
- **Cluster:** Bir veya daha fazla düğümün toplamıdır. Cluster sayesinde tüm verileri içeren indeks oluşturma ve arama kabiliyetleri oluşur. Aynı yapı MongoDB'de de mevcuttur.

- **Shard:** Bir index, çok fazla veri barındırdığı zaman veriyi farklı düğümlere bölmek gerekir. Elasticsearch, bir index'i birden çok parçaya bölmemize olanak tanıyan shard yapısını kullanmaktadır. Bir index oluşturduğunuzda çok kolay bir şekilde kaç tane shard kullanmak istediğinizi tanımlayabilirsiniz. Her bir shard, bir index'in sahip olduğu tüm özelliklere sahip olan ve cluster'daki herhangi bir node'da barındırılabilen tamamen bağımsız bir indexe sahiptir.
- **Replica:** Shard'ın devre dışı kalması ihtimaline karşı index shard'larının bir veya birden çok kopyasının oluşturulabilmesini sağlayan replica shard (replika) yapısı bulunur. Bu yüksek erişilebilirlik için önemlidir. Bu yapı aynı zamanda MongoDB'de de bulunmaktadır.

Elasticsearch'te kullanılan Cluster yapısına bir örnek Şekil 2.8'de verilmiştir.



Şekil 2.8. Elasticsearch cluster örneği.

Elasticsearch, yüzlerce sunucu ve petabaytlarca bilgi üzerinde çalışabilme yeteneğine sahiptir. Ayrıca, doğası gereği dağıtılmış bir sistem ve kaynakları genişletme gücüne sahip olduğundan, bir kümedeki düğümler arasındaki yüklemeleri dengeleme

yeteneği sağlayarak yatay olarak kolayca ölçeklenebildiğinden dolayı daha çok tercih edilmektedir. Bunların yanı sıra Elasticsearch, düğümler arasında oluşabilecek herhangi bir arızadan dolayı veri kaybına yol açabilecek ihtimalleri önlemek için verileri otomaktik olarak çoğaltma gücüne sahiptir. Devasa verileri büyük hızda toplayıp kaydeden, aynı zamanda o verilerin analizini yapan ve onları aranabilir hale getiren uygulama veya sistemlere gereksinim duyan şirketler tarafından son derece tercih edilmektedir. Tam metin arama yapma işinde Elasticsearch'ü kullanan bazı bilişim şirketleri ve platformlar aşağıda verilmiştir (MentorMate, 2017).

- Stack Overflow: Elasticsearch'ü kullanarak tama metin arama, konum belirleme sorguları, kaynakla ilgili sorular veya yanıtları elde etme gibi işlemleri gerçekleştirmektedir.
- Github: Elasticsearch altyapısından yararlanarak arama motoruyla milyarlarca kod satırını sorgulamaktadır.
- Guardian: Editörlere, yayınlanan içerikler hakkında kamuoyu tepkileri (sosyal medya ve ziyaretçi verileri) hakkında güncel geri bildirimler sağlamak için Elasticsearch'ü kullanmaktadır.
- Wikipedia: FTS işleminde Elasticsearch altyapısına dayanarak sonuçlar önerir.

2.11.2.2. Avantajları

- **Otomatik indeks oluşturma:** Diğer sistemlerden farklı olarak Elasticsearch'te veriler yükleme esnasında indekslenir. Veri tabanındaki her bir alanı indeks olarak düşünebiliriz. Veri indeksleme gibi ekstra bir işlem olmadığı için zamandan tasarruf etmiş oluruz.
- **FTS işlemlerini destekleme:** Kullandığı Inverted ve Full Inverted Index sayesinde Elasticsearch, piyasadaki en hızlı FTS işlemini yapan arama motoruna sahiptir.
- **Detaylı sorgulara olanak tanıma:** Elasticsearch, uygulama geliştiricilerinin karmaşık sorgular oluşturmalarına ve bir aramadan en kesin sonuçları alacak şekilde ince ayarlar yapmalarına olanak tanıyan güçlü bir JSON tabanlı DSL'ye (Domain Specific Language - Etki Alanına Özgü Dil) sahiptir. Ayrıca sonuçların sıralanması ve gruplandırılmasını da desteklemektedir.

- **Büyük kapsamda veri tiplerini destekleme:** Elasticsearch, aşağıdakiler gibi yaygın olarak kullanılan tüm veri tiplerini desteklemektedir:
    - **Metin:** `string` (hem yapılandırılmış hem de yapılandırılmamış verilerden oluşabilir)
    - **Sayılar:** `long, integer, short, byte, double, float`
    - **Tarihler:** `date`

  Ayrıca diziler, nesneler, iç içe ve coğrafi veri tipleri, iPV4 ve diğerleri gibi karmaşık veri tiplerini de desteklemektedir.
- **Çeşitli eklentileri (plugins) destekleme:** Elasticsearch, kapasiteleri artırmak için farklı çeşitlerde faydalı eklentiler sunar. Eklentiler, verileri anlamak ve işlemek için daha zengin analizlerin yanı sıra ek güvenlik özelliklerini de sağlamaktadır.
- **Veri organizasyonu sağlama:** Elasticsearch, içinde Ters Dizin (Bkz: Şekil 2.2) tipinde bir tablo oluşturarak belgelerden toplanan tüm benzersiz terimleri oraya doldurur. Aratılan her bir terim oluşturulan Ters dizin tablosuyla karşılaştırılır. Böylece o terimin belgelerin arasında bulunup bulunmadığı çok hızlıca tespit edilir.
- **Hız:** Elasticsearch, karmaşık sorguları son derece hızlı bir şekilde yürütebilir. Ayrıca, sonuç kümesi için bir filtre olarak sık sık kullanılan yapılandırılmış sorguların neredeyse tamamını önbelleğe alarak onları yalnızca bir kez yürütür. Önbelleğe alınmış bir filtre içeren diğer her istek için sonucu önbellekten kontrol eder. Dolayısıyla bu işlem, sorguyu ayrıştırmak ve yürütmek için zamandan tasarruf sağlayarak hızı artırır.

BÖLÜM 3

MATERYAL VE YÖNTEMLER

Bu çalışmada, ikisi RDBMS (MySQL ve MSSQL), diğer ikisi de NoSQL (MongoDB ve Elasticsearch) olmak üzere toplam dört VTYS üzerinde testler gerçekleştirilmiştir. Küçük boyutlu metinler üzerinde testler yapmak için Wikipedia makale özetleri, büyük boyutlu metinler üzerinde testler yapmak için ise Gutenberg'ten alınan kitaplar kullanılmıştır. Her iki veri seti için de seçilen veri tabanlarında ilgili yöntemler ile metin indeksleri oluşturulmuştur. İndekslenmiş veriler üzerinde FTS, indekslenmemiş veriler üzerinde ise normal arama (Like/Regex) yöntemi ile testler yapılmıştır.

MongoDB'de kullanılan klasik metin indeksi B-Tree (B-Ağacı) türündedir. Bulut ortamında çalışan MongoDB Atlas'ta kullanılan "Atlas Search" hizmeti ise Elasticsearch tarafından da kullanılan Apache Lucene kütüphanesini kullanmakta ve bu kütüphanedeki Inverted Index / Full Inverted Index türündeki indekslerle çalışmaktadır.

FTS araması yapabilmek için kullanılan metin indekslerinin oluşturma tarzları tamamen kullanılan veri tabanı sistemine göre değişmektedir. Bu işlem, kimi veri tabanı sisteminde veri yükleme esnasında oluşturulur iken kimisinde ise veriler yüklendikten sonra gerçekleştirilmektedir. İndeksler, sistem ara yüzleri veya komut satırlarından yararlanarak kolayca oluşturulabilmektedir. Bu çalışmada kullanılan veri setlerindeki tüm makale özetleri ve kitaplar İngilizce olduğu için indeksleme yaparken dil seçimi olarak "English" kullanılmıştır.

3.1. MySQL'de Tam Metin Arama

MySQL veri tabanında diğer tüm RDBMS'lerde olduğu gibi SQL'deki LIKE fonksiyonunu kullanarak metin tabanlı aramalar yapmak mümkündür. Fakat bu yöntemle yapılan sorgularda tüm verinin taranması gerekeceği için, veri büyüdükçe sorgu hızı düşer. LIKE yerine indekslenen veriler üzerinde FTS sorguları yaparak daha hızlı sonuç elde etmek mümkündür. FTS sorguları, performansı arttırmak için özel olarak oluşturulmuş bir indeks kullanmaktadır. Bu indeks MySQL'in veri kümesindeki kelimeleri takip etmesini sağlayarak doğal dilde yapılan aramaları kolaylaştırır.

Çoğu ilişkisel veri tabanlarında olduğu gibi MySQL veri tabanında da bir tablo üzerinde FTS sorgusunu yürütmeden önce, o tablonun sorguda kullanılacak ilgili sütunlarının üzerinde tam metin indeksi oluşturmak gerekmektedir. MySQL'de tam metin indeksi oluştumadan önce aşağıda yer alan bazı temel kavramların bilinmesinde fayda vardır (w3resource, 2022):

- FULLTEXT, MySQL'de tam metin indeksi oluşturulmasında kullanılan indeks türüdür;
- Hem InnoDB hem de MyISAM tabloları üzerinde tam metin indeksi oluşturulabilir;
- Tam metin indeksleri yalnızca CHAR, VARCHAR veya TEXT tipindeki sütunlar için oluşturulabilir;
- Büyük veri setleri, tam metin indeksi olan tablolara daha hızlı eklenebilir;
- Performans açısından veri setini yükleme tam metin indeksinin oluşturulmasından önce yapılmalıdır.

3.1.1. Tam metin indeksi oluşturma

MySQL'de tam metin indeksi `CREATE TABLE`, `CREATE INDEX` ve `ALTER TABLE` komutları ile oluşturulabilir:

- `CREATE TABLE` ile tablo oluşturulurken tam metin indeksinde yer alacak olan sütun veya sütunlar FULLTEXT fonksiyonunda belirtilir.

```
CREATE TABLE tabloAdı (
     sütun_1, sütun_2, ...,
     FULLTEXT (sütun_1, sütun_2, ...)
)
```

- `ALTER TABLE` ile daha önce oluşturulmuş olan bir tablo üzerinde tam metin indeksi oluşturulabilir.

```
ALTER TABLE tabloAdı
ADD FULLTEXT (sütun_1, sütun_2, ...)
```

- `CREATE INDEX` bir veya daha fazla sütun için adlandırılmış bir tam metin indeksi oluşturulmasında kullanılır.

```
CREATE FULLTEXT INDEX indeksAdı
ON tabloAdı (idx_sütunAdı,...)
```

Oluşturulan FULLTEXT tipindeki indeksleri silmek için ise normal indeksleri silmek için kullanılan `ALTER TABLE tabloAdı DROP INDEX indeksAdı` kullanılabilir.

3.1.2. Oluşturulan indeks üzerinde arama yapma

MySQL'de indekslenmiş bir sütunun ve sütunların üzerinde FTS işlemini uygulamak için WHERE ifadesinden sonra MATCH ve AGAINST fonksiyonları kullanılır:

- `MATCH(sütun_adları) AGAINST('aranan kelime' arama yöntemi)`
  MATCH fonksiyonun içinde belirtilen sütunlarda AGAINST içinde belirtilen aranan kelimeyi yine burada belirtilen arama yöntemine dayanarak elde eder.

Oluşturulan bir tam metin indeksi üzerinde FTS işlemi yapmak için genellikle üç farklı yöntemlerden biri kullanılır:

- **Doğal Dil Tam Metin Aramaları (Natural Language Full-Text Searches):** Varsayılan arama yöntemi olduğu için hiçbir yöntem belirtilmediğinde veya `IN NATURAL LANGUAGE MODE` yazıldığında kullanılan yöntemdir (Vileikis, 2020). Bu yöntemde yapılan aramalar varsayılan olarak büyük/küçük harfe duyarlı olmayan biçimde gerçekleşmektedir. Büyük/küçük harfe duyarlı bir tam metin araması yapmak için indekslenmiş sütunlarda büyük/küçük harfe duyarlı veya ikili seçeneği kullanılmalıdır. Örneğin, utf8mb4 karakter kümesini kullanan bir sütunda, tam metin aramalarını büyük/küçük harfe duyarlı hale getirmek için `utf8mb4_0900_as_cs` veya `utf8mb4_bin` seçeneği kullanılabilir (w3resource, 2022).

```
SELECT * FROM tabloAdı
WHERE MATCH (sütun_1, sütun_2, ...)
AGAINST ('aranan_kelime' IN NATURAL LANGUAGE MODE)
```

- **Boole Tam Metin Aramaları (Boolean Full-Text Searches):** `IN BOOLEAN MODE` yöntemi ile arama yapıldığında, belirli karakterlerin arama dizesindeki kelimelerin başında veya sonunda özel anlamları olur. Bir kelimenin başında '+' olursa o kelimenin bulunduğu satırlar, '-' olursa o kelimenin bulunmadığı satırlar aranmış olur. Örneğin aşağıdaki kod paçası "Covid19" kelimesini içermeyen ve "Dunya" kelimesi ile başlayan ve en az bir kelime içeren kayıtları bulur. Her bir sonuç "insan" ifadesini içerebilir veya içermeyebilir.

```
SELECT * FROM tabloAdı
WHERE MATCH (sütunAdı)
AGAINST ('+Dunya* insan –Covid19' IN BOOLEAN MODE)
```

- **Sorgu Genişletme Aramaları (Query Expansion Searches):** Bu yönteme ayrıca "Genişletilmiş Aramalar" da denir. Burada, kelime araması iki kere yapılmaktadır. İlkinde aranan kelime için indeks üzerinden taramalar yapılır. Aranan ilk kelime ile bulunan kayıtlar için bir daha arama yapılır. İkinci aramada sonradan bulunan ifadeler üzerinden olduğundan dolayı doğal olarak geri döndürülen sonuçlar daha fazla olur (Walker, 2021). Örneğin "database" ifadesi

aratıldığı zaman “MySQL” ifadesi konuyla alakalı olduğundan geri döndürülebilir. İkinci aramada bu defa arama işlemi “MySQL” üzerinden yapılarak ilgili tüm kayıtları geri döndürür. Bu işlem `WITH QUERY EXPANSION` ifadesiyle yapılmaktadır.

```
SELECT * FROM tabloAdı
WHERE MATCH (sütunAdı)
AGAINST ('database' WITH QUERY EXPANSION)
```

MySQL’de yapılan FTS işlemlerinde yukarıda bahsi geçen yöntemlerden `IN BOOLEAN MODE` tercih edilmiştir.

3.1.3. FTS yapılandırması

MySQL, arama biçimini ayarlamamıza izin veren birkaç FTS yapılandırma seçenekleriyle birlikte gelir. En önemlilerinden birkaç tanesi aşağıda yer almaktadır (Walker, 2021):

- `innodb_ft_min_token_size:` İndekse dahil edilmiş kelimeler için minimum karakter uzunluğunu ayarlar. Bu değerden daha az karaktere sahip kelimeler indekse eklenmez. Varsayılan değer üç karakterdir ve “a”, “an” ve “I” gibi son derece yaygın sözcükleri hariç tutar. Bu ayar InnoDB tabloları için geçerli olup MyISAM için `ft_min_word_len` kullanılmaktadır.
- `innodb_ft_max_token_size:` İndekse eklenen kelimelerin maksimum uzunluğunu ayarlar. Daha uzun kelimeler aranamaz. MyISAM tabloları için `ft_max_word_len` kullanılmaktadır.
- `innodb_ft_enable_stopword:` Varsayılan olarak açık olan bu ayar, MySQL'in etkisiz kelimeleri filtreleyip filtrelemeyeceğini kontrol etmesini sağlar. Etkisiz kelimeler, arama sonuçlarını gereksiz yere etkileyebilecek çok yaygın olarak kullanılan kelimelerdir. Varsayılan etkisiz kelime listesi 36 sık kullanılan kelimeyi içermektedir.
- `innodb_ft_user_stopword_table:` Bir veri tabanında VARCHAR tipinde olan ve adlandırılmış bir tablo içinde kendi etksiz kelime havuzumuzu da oluşturabiliriz.

Yukarıdaki ayarlar genellikle MySQL sunucu yapılandırma dosyasında yer alır. (Linux üzerinde genellikle bu dosya: /etc/mysql/my.cnf). Ayarları değiştirdikten sonra MySQL hizmetini yeniden başlatmak ve tablolar üzerinde tam metin indekslerini yeniden oluşturmak gereklidir. Aksi takdirde, daha önce indekslenen veriler kullanılmaya devam edilir. Bir InnoDB tablosunu yeniden indekslemek için `OPTIMIZE TABLE tabloAdı` kodu kullanılmaktadır. MyISAM tabloları için `REPAIR TABLE tabloAdı QUICK` kodu kullanılmaktadır (Walker, 2021).

3.2. Microsoft SQL Server'da Tam Metin Arama

FTS işlevi, SQL Server 2008 sürümünden itibaren MSSQL veri tabanı motoruna dahil edilmiştir. Bir veri tabanında FTS işlemi yapmak için SQL Full-text Filter Daemon Launcher (MSSQLFDLauncher) hizmetinin komut satırı penceresinden veya Hizmetler kısmından çalışır hale getirilmesi gerekmektedir. Bu hizmet sayesinde filtreleme[1] ve stemming işlemlerinden sorumlu olan Filter Daemon Host (fdhost.exe) prosesleri oluşturulur. Burada bahsedilen filtreleme etkisiz kelime filtrelemesinden farklıdır. Etkisiz kelime fitrelemesi SQL Server prosesi (sqlservr.exe) tarafından gerçekleştirilmektedir (Elias, 2017).

MSSQL veri tabanında FTS ile ilgili bazı faydalı sorgulara, indeks oluşturma ve sorgu yapma işlemlerine takip eden alt bölümlerde yer verilecektir.

3.2.1. FTS özelliğinin veri tabanında kullanılabilir hale getirilmesi

FTS özelliğinin sunucuda yüklü olup olmadığını kontrol etmek için aşağıdaki sorgu kullanılabilir:

```
SELECT
     CASE FULLTEXTSERVICEPROPERTY('IsFullTextInstalled')
          WHEN 1 THEN 'Full-Text installed.'
          ELSE 'Full-Text is NOT installed.'
     END
```

[1] Filtreleme: Word, Excel, Xml gibi farklı biçimlerde saklanmış olan belgeler için türlerine göre farklı filtreler kullanılır. Filtre, belgedeki etiketler gibi gömülü biçimlendirme öğelerini kaldırarak salt metni elde eder.

FTS özelliğinin mevcut veri tabanında kullanılabilir durumda olup olmadığını öğrenmek için aşağıdaki sorgu kullanılabilir:

```
SELECT is_fulltext_enabled
FROM sys.databases
WHERE database_id = DB_ID()
```

FTS özelliğini kullanılabilir hale getirmek için ise aşağıdaki prosedür yürütülür:

```
exec sp_fulltext_database 'enable'
```

3.2.2. Tam metin kataloğu oluşturma

Tam metin indekslerinin saklandığı bir sanal veri tabanı nesnesi olan tam metin kataloğunun olup olmadığını kontrol etmek için aşağıdaki sorgu yazılabilir:

```
SELECT * FROM sys.fulltext_catalogs
```

Eğer mevcut bir katalog yoksa bir tane oluşturup varsayılan yapılmalıdır. Aşağıdaki kod `FullTextCatalog` isminde bir tam metin arama kataloğu oluşturur ve varsayılan katalog yapar:

```
CREATE FULLTEXT CATALOG FullTextCatalog AS DEFAULT
```

3.2.3. Tam metin indeksi oluşturma

Tam metin indeksinin oluşturulması MSSQL Server ara yüzü olan SSMS ile veya aşağıdaki gibi bir kod ile yapılabilir (Örnekteki ‘PK_tabloAdı’ ilgili tablonun birincil anahtarının ‘constraint’ ismidir):

```
CREATE FULLTEXT INDEX ON tabloAdı (
      sütunAdı LANGUAGE English
) KEY INDEX PK_tabloAdı
WITH
   CHANGE_TRACKING = AUTO,
   STOPLIST = OFF
```

- `LANGUAGE English:` Bu ifade ile belirtilen sütun üzerinde İngilizce diline göre FTS indeksi oluşturulur. İndeks oluşturulmasında varsayılan dil İngilizce olduğundan yazılmasa da aynı türde indeks oluşur. `English` yerine kodu olan `1033` yazılabilir. `Turkish` veya `1055` yazarak indeksleme dilini Türkçe olarak tanımlayabiliriz. Dil ismi veya kodu yazmak yerine sıfır (`0`) yazılırsa dilden bağımsız indeks oluşturmuş oluruz. İndeks oluşturmada kullanılan 52 dili `SELECT * FROM sys.fulltext_languages ORDER BY lcid` sorgusu ile kolayca öğrenebiliriz (Elias, 2017).
- `WITH:` Bu komut sayesinde oluşturulan indekse seçenekler tanımlanabilir.
- `CHANGE_TRACKING = AUTO`: bu komut ile DML işlemleri sonrası indeksin otomatik olarak güncellenmesi sağlanır. Yazılmadığı taktirde varsayılan olarak aynı işlem yapılır.
- `STOPLIST = OFF:` Bu komut ile etkisiz kelime filtrelemesi göz ardı edilir.

Oluşturulan indeksin doldurulması (populating) zaman almaktadır. İndeksin hangi oranda doldurulduğunu öğrenmek için SSMS ara yüzünde ilgili veri tabanının altındaki **Storage** düğümü **Full Text Catalogs** altında bulunan oluşturduğumuz kataloğa çift tıklayarak detaylı bilgilere ulaşabiliriz. İndeksin tam dolmamış olması yazılan sorgunun çalışmasına engel olamaz. Fakat sorgu esnasında eksik sonuçlar elde edilebilir.

3.2.4. Oluşturulan indeks üzerinde arama yapma

MSSQL Server'da oluşturulan indeksler üzerinde FTS işlemi iki farklı fonksiyon ile yapılabilir:

- **CONTAINS** fonksiyonu ile yapılan aramalarda, aranan kelime indekslenen sütunlar üzerinde tam olarak yazıldığı hali ile aranır.

```
SELECT * FROM tabloAdı
WHERE CONTAINS(sütunAdı, 'arananKelime')
```

Benzer sorgu LIKE ile FTS olmaksızın yapılırsa kelimenin ön ve son ekleri de dikkate alınacağı için daha fazla sonuç elde edilebilir. Örneğin "like" kelimesi aratıldığı zaman "likely", "unlike" veya "dislike" gibi kelimeleri içeren kayıtlar da bulunabilir. 'arananKelime' kısmına "Trakya University" gibi birden fazla kelime içeren bir ifade yazılırsa hata verir. Sorgular ilk denemede yavaş çalışsa da tekrarlanması halinde *RAM*'e çıkarılan indeksler sayesinde daha hızlı sonuç elde edilebilir.

- **FREETEXT** fonksiyonu ile aranan kelime indekslenen sütunlar üzerinde etkisiz kelimelerin elenmesi ve stemming işlemleri uygulanarak aranır.

```
SELECT * FROM tabloAdı
WHERE FREETEXT(sütunAdı, 'arananKelime')
```

Örneğin 'arananKelime' kısmına "likes", "liking" veya "likely" kelimeleri yazılırsa, hepsi "like" kökü ile aranacağı için aynı sonuç elde edilecektir. Aynı zamanda iki veya daha fazla kelimeden oluşan bir ifade aranabilir. Fakat aranan kelimeler bütünleşik bir ifade olarak aranmaz, herhangi birini içeren kayıtlar getirilir. Yani "Trakya University" aradığımızda "Trakya" veya "University" içeren kayıtları elde etmiş oluruz.

3.3. MongoDB'de Tam Metin Arama

Bulut ortamında barındırılan veriler üzerinde FTS işleminin yapılması için MongoDB tarafında sunulan "MongoDB Atlas Search" kullanılmaktadır. Bu çalışmadaki veriler üzerinde FTS işlemi yapmak için ise MongoDB tarafından kullanılan klasik metin indeksi olan B-Tree kullanılmıştır. Bunun sebebiyse verilerin yerel bir sunucuda saklanmış olmasıdır.

3.3.1. Metin indeksi oluşturma

MongoDB'de metin indeksi oluşturmak için **createIndex** yöntemi aşağıdaki gibi kullanılır:

```
db.koleksiyonAdı.createIndex({alanAdı: "text"})
```

Birden fazla alan üzerinde metin indeksi oluşturmak istenirse, alanların aralarına virgül konularak aşağıdaki gibi yazılabilir:

```
db.koleksiyonAdı.createIndex({alanAdı_1: "text",
                              alanAdı_2: "text"})
```

Koleksiyondaki tüm string veri türündeki alanların indekse dahil olması için "$**" ifadesi kullanılabilir.

```
db.koleksiyonAdı.createIndex({"$**": "text"})
```

Diğer birçok veri tabanı yönetim sisteminde olduğu gibi MongoDB'de de etkisiz kelimelerin elenmesi ve filtrelenmesi, indeks oluşturma esnasında seçilen dile dayanmaktadır. Oluşturulan indeksler için dil seçimi yapılmazsa varsayılan olarak İngilizce dili kullanılmaktadır. Aşağıda verilen kodda, indeksleme dili Türkçe olan bir metin indeksinin MongoDB'deki oluşturma şekli gösterilmiştir:

```
db.koleksiyonAdı.createIndex (
   { alanAdı: "text" },
   { default_language: "turkish" }
```

3.3.2. Oluşturulan indeks üzerinde arama yapma

MongoDB'de metin indeksi üzerinde arama yapma işlemi MSSQL Server'daki FREETEXT yöntemine benzer. İndeksleme esnasında sadece kelimelere ait kökler saklanır. Birden fazla kelime araması yapılırsa hepsi ayrı ayrı aranır ve elde edilen toplam sonuç getirilir. MongoDB'de etkisiz kelimeler indekste yer almadığından dolayı bu tip bir kelime aranırsa sonuç boş döner. Fakat MSSQL'de bu kelimeleri indekse dâhil edebilme şansımız bulunmaktadır.

MongoDB'de metin indeksi üzerinde arama yapmak için `find` yöntemi içinde `{$text:{$search: …}}` sözdizimi kullanılır.

```
db.koleksiyonAdı.find({$text:{$search: "arananKelime"}})
```

3.4. Elasticsearch'te Tam Metin Arama

Elasticsearch, HTTP (Hyper Text Transfer Protocol - Hiper Metin Transfer Protokolü) Web ara yüzü ve şemadan bağımsız JSON belgeleriyle dağıtılmış, tam metin arama motoru sağlamaktadır. Python, .Net ve C++ gibi birçok yazılım diliyle kullanılabilir. Daha önce bahsedildiği gibi Elasticsearch'te indeks oluşturmak diye bir işlem yoktur. Dolaysıyla veriler üzerinden doğrudan FTS işlemi yapılmaktadır. Bu çalışmada FTS işlemini yapmak için Python dili tercih edilmiştir. Elaticsearch'te veriler üzerinde metin araması yapmak için `search` yöntemi içinde `{"db_adı", "match": {"alan_adı": …}}` sözdizimi kullanılır.

```
es = Elasticsearch()
es.search(index="db_adı", body={"query": {"match":
{"alan_adı": "arananKelime"}}})
```

BÖLÜM 4

TEZ KAPSAMINDA SEÇİLEN VERİ TABANI YÖNETİM SİSTEMLERİNİN PERFORMANS KARŞILAŞTIRMASI

Bu bölümde, tez çalışmasında kullanılan test ortamları ve seçilen veri tabanlarına veri setlerinin yüklenmesi için yapılan tüm ayarlardan bahsedilmiştir. Ayrıca FTS işlemleri ve normal aramalara yönelik veri tabanları üzerinde yapılan tüm test sonuçları da verilmiştir. Tüm testlerde, veri tabanlarının her birinden üçer defa sonuç alınarak elde edilen ortalama arama süresi değerleri grafik olarak verilmiştir. Yapılan testlerin sonuçlarına ait bilgiler EKLER kısmında yer almaktadır.

4.1. Kullanılan Cihazın Özellikleri

Çalışmada seçilen veri tabanlarının yüklendiği ve yapılan tüm testlerde kullanılan cihazın özellikleri aşağıda verilmiştir:

- ✓ **İşletim Sistemi:** Windows 10 Enterprise / 64 bit
- ✓ **İşlemci:** Intel(R) Xeon(R) CPU E3-1225 v3 @ 3.20GHz 3.20 GHz
- ✓ **RAM:** 16,0 GB
- ✓ **Sabit Disk:** 480 GB SSD

4.2. Test için Kullanılan Veri Setleri

Testlerin yerel bir bilgisayarda yapılması gerektiği için her bir veri tabanı sistemine büyük metinlerden ve küçük metinlerden oluşan toplam iki veri seti yüklenmiştir:

- **Gutenberg veri seti:** https://www.gutenberg.org/ sitesinden alınan toplam 9.951.064.264 bayt boyutunda 21.405 adet İngilizce kitaptan oluşmaktadır.
- **Wikipedia veri seti:** https://www.dbpedia.org/ sitesinden alınan toplam 2.317.910.604 bayt boyutunda 4.383.126 adet İngilizce makale özetinden oluşmaktadır (bu özetler 114 adet farklı dosyada bulunmaktadır).

İki farklı veri seti kullanılmasındaki amaç; FTS türü aramalarda 'az sayıda & büyük boyutlu' dokümanların olması ile 'çok sayıda & küçük boyutlu' dokümanların olması durumları arasındaki farkı ortaya koymaktır. Gutenberg veri setinde ortalama doküman boyutu 464.894 bayt iken, Wikipedia veri setinde bu değer 529 bayttır.

Veri setlerinin 'txt' veya 'json' dosya biçimlerinde saklanan kopyalarından veri tabanlarına uygun bir şekilde aktarılması için bazen ilgili veri tabanının import aracından, bazen de Python yazılım dilinden yararlanmıştır.

4.3. Testlerde Kullanılan Veri Tabanı Sistemleri

Bu tez çalışmasında, iki adet RDBMS (MySQL ve Microsoft SQL Server) ve iki adet NoSQL (MongoDB ve Elasticsearch) olmak üzere FTS aramayı destekleyen dört farklı veri tabanı yönetim sistemi üzerinde arama testleri yapılmıştır. Tercih edilen tüm veri tabanları *db-engines.com* sitesine göre en popüler 10 veri tabanı içinde yer almaktadır. Tezin yazıldığı 2022 yılı başlarında MySQL ikinci, MSSQL Server ise üçüncü en popüler veri tabanıdır. En çok kullanılan belge deposu türündeki MongoDB genel sıralamada beşinci ve en çok kullanılan arama motoru türündeki Elasticsearch ise genel sıralamada sekizinci sıradadır.

Seçilen veri tabanı sistemlerinin testlerde kullanılan versiyonları aşağıdaki gibidir:

- MySQL 8.0.24-commercial (64-bit)
- Microsoft SQL Server Developer 14.0.2037.2 (64-bit)
- MongoDB 4.2.2 (64-bit)
- Elasticsearch 7.13.1 (64-bit)

4.3.1. MSSQL'de yapılan test öncesi işlemler

4.3.1.1. Veri tabanı oluşturma

MSSQL'de Gutenberg veri seti için oluşturulan veri tabanına 'Gutenberg' ve verileri saklayacak olan tabloya 'books' ismi verilmiştir. Bu tabloda aşağıdaki iki alan vardır:

- `name: char` tipinde bir sütun olup kitapların dosya isimlerini saklamaktadır. Aynı zamanda birincil anahtardır.
- `content: varchar` tipinde bir sütun olup kitapların içerik kısmını saklamaktadır. FTS işlemleri bu sütun üzerinden yapılmaktadır.

MSSQL'de Wikipedia veri seti için oluşturulan veri tabanına 'Wikipedia' ve verileri saklayacak olan tabloya 'articles' ismi verilmiştir. Bu tabloda aşağıdaki iki alan vardır:

- `id: int` tipinde bir sütun olup `COUNT()` gibi sayısal işlemlerde kullanmak amacıyla oluşturulmuştur.
- `abstract: varchar` tipinde bir sütun olup makale özetlerini metin halinde saklamaktadır. FTS işlemleri bu sütun üzerinden yapılmaktadır.

4.3.1.2. Veri seti yükleme

MSSQL veri tabanına Gutenberg veri setini JSON biçimindeki "gutenberg.json" isimli dosyadan yüklemek için çalıştırılan SQL kodu aşağıda verilmiştir:

```
SELECT book.* INTO books
FROM OPENROWSET (BULK 'C:\gutenberg.json', SINGLE_CLOB) AS j
CROSS APPLY OPENJSON(BulkColumn)
WITH(name char(15), content varchar(max)) AS book
```

Wikipedia veri setine ait İngilizce makale özetleri 114 adet ".txt" uzantılı düz metin dosyasının içinde yer almaktadır. Bu özetlerin MSSQL veri tabanına yüklenmesi için kullanılan Python kodunun ilgili bölümü aşağıda verilmiştir:

```
for i in range(114):
dosya = open("en" + str(i) + ".txt", encoding = "utf-8")
string = dosya.read()
    dosya.close()
    satirlar = string.split(sep="\n")
for s in satirlar:
   try:
      cur.execute("INSERT INTO en(abstract) VALUES(?)", (s))
   except:
      print('eklenemeyen özet : \n' + s)
      pass
con.commit()
```

4.3.1.3. İndeks oluşturma

Gutenberg veri seti üzerinde oluşturulan metin indeksi aşağıda gösterildiği gibi `content` adı verilen sütun üzerindedir:

```
CREATE UNIQUE INDEX uniq_id_fts ON dbo.books(ID);
CREATE FULLTEXT CATALOG FTS AS DEFAULT;
CREATE FULLTEXT INDEX ON books(content)
   KEY INDEX uniq_id_fts ON FTS
   WITH STOPLIST = SYSTEM;
```

Wikipedia veri seti üzerinde oluşturulan metin indeksi aşağıda gösterildiği gibi `abstract` adı verilen sütun üzerindedir:

```
CREATE UNIQUE INDEX uniq_id_fts ON dbo.en(id);
CREATE FULLTEXT CATALOG FTS_Wikipedia AS DEFAULT;
CREATE FULLTEXT INDEX ON articles(abstract)
   KEY INDEX uniq_id_fts ON FTS_Wikipedia
   WITH STOPLIST = SYSTEM;
```

Her iki veri seti için oluşturulan indekslere ait detaylı bilgiler Çizelge 4.1'de verilmiştir.

Çizelge 4.1. MSSQL Server'da oluşturulan FTS indekslerinin detaylı bilgileri.

Özellik	Gutenberg Veri Seti	Wikipedia Veri Seti
Boyut (Catalog Size)	3329 MB	1377 MB
Eleman/Satır Sayısı (Item Count)	21.405	4.383.126
Anahtar Sayısı (Unique Key Count)	5.184.036	4.484.223
Oluşturulma Süresi	15 dakika	8 dakika

4.3.2. MySQL'de yapılan test öncesi işlemler

4.3.2.1. Veri tabanı oluşturma

MySQL üzerinde de MSSQL'de kullanılan veritabanı, tablo ve alan isimleri kullanılmıştır. Gutenberg veri seti için tek fark `content` alanının `longtext` türünde olmasıdır. Wikipedia veri seti için tek fark ise `abstract` alanı `text` türündedir.

4.3.2.2. Veri seti yükleme

MySQL'e Gutenberg veri setinin yüklenmesi işlemi, daha önce MSSQL'de yaratılan tablodan verileri okuyan ve MySQL'deki ilgili tabloya aktaran aşağıdaki Python kodu ile yapılmıştır:

```
mscur = msdb.cursor()
mycur = mydb.cursor()
mscur.execute("SELECT name FROM Books")
kitapisimleri = mscur.fetchall()
for kitap in kitapisimleri:
   mscur.execute("SELECT content FROM Books WHERE name = ?",
                                                    kitap[0])
   icerik = mscur.fetchone()
   mycur.execute('INSERT INTO books VALUES (%s, %s)',
                                        (kitap[0], icerik[0]))
mydb.commit()
```

MySQL varsayılan olarak 4 MB boyutundan büyük olan verilerin aktarılmasında hata verdiği için ve bazı kitapların boyutu bu değerden büyük olduğu için kodu çalıştırmadan önce MySQL'in `my.ini` dosyasında izin verilen en büyük paket boyutunu 10MB yapmamız gerekti (`max_allowed_packet=10M` [varsayılan değer 4M idi]). MySQL'e Wikipedia veri setinin yüklenmesinde yapılan işlemler ise MSSQL'deki ile aynıdır.

4.3.2.3. İndeks oluşturma

MSSQL'de de olduğu gibi, burada Gutenberg veri seti üzerinde oluşturulan metin indeksi için `content` adı verilen sütundan yararlanmıştır.

```
CREATE FULLTEXT INDEX Gutenberg ON books(content)
```

Wikipedia veri seti üzerinde oluşturulan metin indeksi için `abstract` adı verilen sütundan yararlanmıştır.

```
CREATE FULLTEXT INDEX Wikipedia ON articles(abstract)
```

Her iki veri seti için oluşturulan indekslere ait detaylı bilgiler Çizelge 4.2'de verilmiştir.

Çizelge 4.2. MySQL'de oluşturulan FTS indekslerinin detaylı bilgileri.

Özellik	Gutenberg Veri Seti	Wikipedia Veri Seti
Boyut (Index Size)	4.126,72 MB	1.474,56 MB
Tablo Satırları (Table Rows)	21.483	4.172.119
Oluşturulma Süresi	1 saat 16 dakika	15 dakika

4.3.3. MongoDB'de yapılan test öncesi işlemler

4.3.3.1. Veri tabanı oluşturma

MongoDB'de Gutenberg veri seti için oluşturulan veri tabanına 'Gutenberg' ve verileri saklayacak olan koleksiyona 'books' ismi verilmiştir. Koleksiyondaki dokümanlar aşağıdaki alanlarda yer almaktadır:

- `_id: ObjectId` tipinde bir alan olup MongoDB tarafından otomatik olarak oluşturulmaktadır.
- `name: string` tipinde bir alandır. Kitapların dosya isimlerini saklamaktadır. FTS işlemleri bu alan üzerinden yapılmaktadır.
- `content: string` tipinde bir alandır. Kitapların içerik kısmını saklamaktadır.

MongoDB'de Wikipedia veri seti için oluşturulan veri tabanına 'Wikipedia' ve verileri saklayacak olan koleksiyona 'articles' ismi verilmiştir. Koleksiyondaki dokümanlar aşağıdaki alanlarda yer almaktadır:

- `_id: ObjectId` tipinde bir alan olup MongoDB tarafından otomatik olarak oluşturulmaktadır.
- `abstract: string` tipinde bir alan olup makale özetlerini metin halinde saklamaktadır. FTS işlemleri bu alan üzerinden yapılmaktadır.

4.3.3.2. Veri seti yükleme

Gutenberg veri setini MongoDB'ye yüklemek için json üzerinden veri aktarımı yapabilen *mongoimport* aracı kullanılmıştır. MSSQL'e yüklemek için de kullanılan "gutenberg.json" dosyasından aşağıdaki komut ile aktarım sağlanmıştır:

```
mongoimport -d Gutenberg -c books --file gutenberg.json
```

Wikipedia veri setinin yüklenmesinde kullanılan Python kodu MSSQL için kullanılan koda benzemektedir. Farklı olan 'try:' altında 'cur.execute(…)' ifadesi yerine aşağıda verilen satırın kullanılmış olmasıdır:

```
collection.insert_one({ "abstract": s })
```

4.3.3.3. İndeks oluşturma

MongoDB'de Gutenberg veri seti üzerinde metin indeksi oluşturmak için `content` alanından yararlanmıştır:

```
db.books.createIndex({content: "text"})
```

Wikipedia veri seti üzerinde oluşturulan metin indeksi için aşağıda verilen `abstract` alanı kullanılmıştır:

```
db.articles.createIndex({abstract: "text"})
```

Her iki veri seti için oluşturulan indekslere ait detaylı bilgiler Çizelge 4.3'te verilmiştir.

Çizelge 4.3. MongoDB'de oluşturulan FTS indekslerinin detaylı bilgileri.

Özellik	Gutenberg Veri Seti	Wikipedia Veri Seti
Boyut (Index Size)	1491.2 MB	1584.8 MB
Eleman/Satır Sayısı (Count)	21405	4383126
Ortalama Nesne Boyutu (Average Object Size)	465168	563
Oluşturulma Süresi	19 dakika	14 dakika

4.3.4. Elasticsearch'de indeks oluşturma ve veri seti yükleme

Elasticsearch bir arama moturu (search engine) olduğu için diğer üç veri tabanından farklı şekilde çalışmaktadır. Veri tabanı ve altında Tablo veya Koleksiyon gibi yapılar yoktur. Elasticsearch zaten tam metin aramaya yönelik geliştirildiği için yüklenen her veriyi aslında Inverted Index türünde bir indeks içinde saklamaktadır. Elasticsearch'te Gutenberg veri seti için 'gutenberg', Wikipedia veri seti için ise 'wikipedia' isminde birer indeks oluşturulmuştur.

Elasticsearch'e Gutenberg veri setini "gutenberg.json" dosyasından yüklemeyi denediğimizde çok uzun sürdüğü için, kitapların içeriğinin bulunduğu ".txt" uzantılı dosyalar üzerinden Python progamı ile kopyalama yapılmıştır. Aşağıda Elasticsearch altında yarattığımız 'gutenberg' isimli indekse Gutenberg veri setinin yüklenmesini sağlayan Python kodu verilmiştir:

```
from elasticsearch import Elasticsearch
from os import listdir, chdir
es = Elasticsearch()
chdir("C:\\Data\\Gutenberg")
for i in listdir():
    dosya = open(i, encoding = "utf8",  errors="surrogateescape")
    kitap = dosya.read()
    dosya.close()
    doc = { 'name': i, 'content': kitap }
    res = es.index(index="gutenberg", body=doc)
    if res['result'] != 'created': print(i, res)
```

İndekse eklenen dokümanlarda `name` ve `content` isminde 'text' veri türünde iki alan olup ilkinde kitapların dosya isimleri, ikincisinde kitaplara ait içerikler saklanmaktadır.

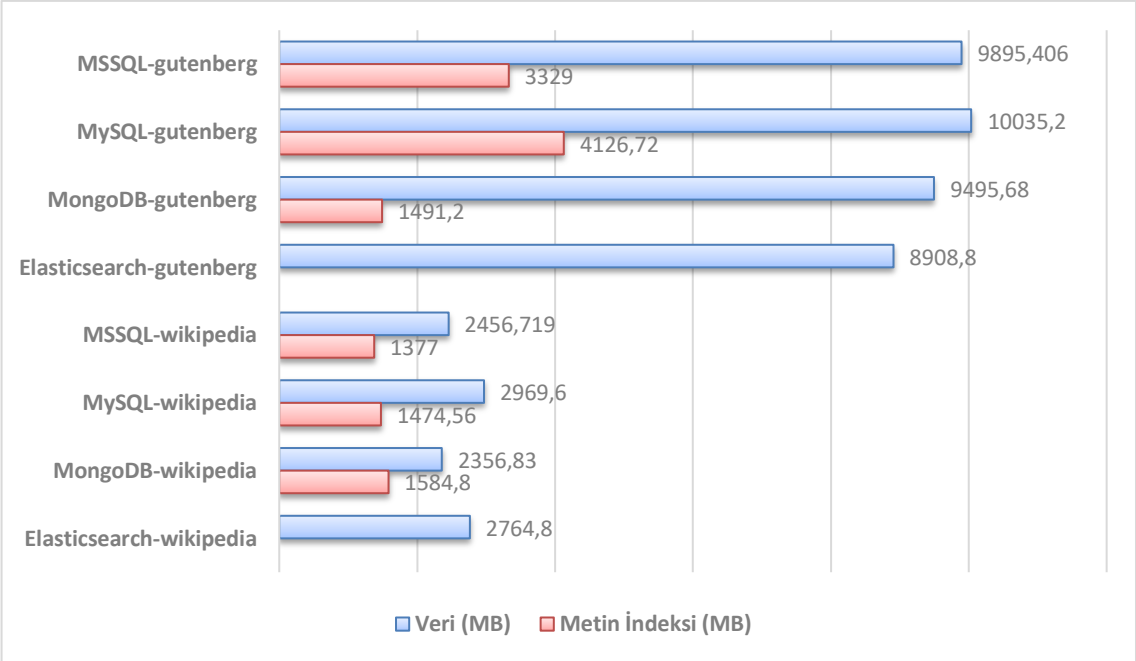
Elasticsearch'e Wikipedia veri setinin yüklemesi için kullanılan Python kodu MSSQL, MySQL ve MongoDB için kullanılan koda benzerdir. Sadece 'try:' bloğu içindeki kısım aşağıdaki gibidir:

```
doc = { 'id': id, 'abstract': s }
res = es.index(index="wikipedia", body=doc)
id += 1
```

Bu veri setine eklenen dokümanlardaki `id` alanı 'long' veri türünde bir alan iken, `abstract` ise 'text' veri türünde bir alan olup makalelere ait özetleri saklamaktadır.

4.3.5. Veri ve indeks boyutlarının karşılaştırılması

Teste dahil olan veri tabanlarında saklanan verilerin ve oluşturulan metin indekslerinin boyutlarının karşılaştırılması Şekil 4.1'de verilmiştir. Elasticsearch'te saklanan veriler doğrudan indeks içinde saklandığı için sadece bir değer verilmiştir (aslında bu değer hem veri hem de indeks olarak kabul edilebilir).



Şekil 4.1 Saklanan verilerin ve metin indekslerinin boyutlarının karşılaştırılması.

Şekil 4.1'deki sonuçlara göre, Gutenberg veri seti için en küçük boyutlu metin indeksini MongoDB, Wikipedia veri seti için ise MSSQL yaratmıştır. MySQL'in oluşturduğu metin indeksleri her iki veri seti için de MSSQL'den fazla yer kaplamaktadır. Gutenberg veri seti için her iki ilişkisel veri tabanından çok daha küçük boyutlu indeks yaratan MongoDB, Wikipedia veri seti için her ikisinin de gerisinde kalmıştır. Elasticsearch'e ait değerleri "veri" olarak değerlendirirsek Gutenberg için en iyi, "indeks" olarak değerlendirirsek her iki veri seti için de en kötü sonuçları verdiği söylenebilir.

Sonuçları VT kategorilerine göre değerlendirmek gerekirse, hem Gutenberg veri seti hem de Wikipedia veri seti için NoSQL VT'leri arasında indeks boyutu en fazla olan Elasticsearch olurken RDBMS VT'leri arasında MySQL olmuştur. İndeks oluşturma sürelerinin karşılaştırılmasını gösteren sonuçlar Çizelge 4.4'te verilmiştir.

Çizelge 4.4. Oluşturulan indekslerde süre karşılaştırılması.

Veri Tabanı Yönetim Sistemleri	Gubenberg Veri Seti	Wikipedia Veri Seti
MySQL	1 saat 16 dakika	15 dakika
MongoDB	19 dakika	14 dakika
MSSQL	15 dakika	8 dakika

4.4. Yapılan Sorgu Testlerinin Sonuçları

Seçilen VT'lerden MySQL, MSSQL ve MongoDB üzerinde oluşturulan metin indeksleri kullanılarak FTS sorguları ve indeks kullanılmadan Like/Regex sorguları yapılarak iki farklı şekilde performans ölçülmüştür. Elasticsearch üzerinde ise yapısı gereği sadece FTS türünde sorgu testleri yapılmıştır.

Bir VT üzerinde testler yapılırken diğer VT'ler kapalı durumda tutularak işlemci ve hafızanın maksimum verim ile kullanılması sağlanmıştır. Hızlı sonuç alınan FTS aramaları için 100 farklı kelime, yavaş sonuç alınan Like/Regex sorguları için ise 10 farklı kelime Python ile yazılan kodlar ile arka arkaya aranmıştır. FTS ve Like/Regex aramalarında kullanılan kelimeler sırasıyla Çizelge 4.5 ve Çizelge 4.6'da verilmiştir.

Çizelge 4.5. FTS aramalarında kullanılan kelimeler.

allows	proposes	characteristics	Germany	Norway
computer	seasons	religious	Spain	Sweden
original	handles	artist	eventually	anthem
turns	grand	ocean	champion	broadcast
staff	England	farming	Mexico	Jerome
feels	conclusion	achievement	Egypt	Hannah
fired	retiring	overlooking	Turkey	phone
attachment	judgment	student	grandmother	rehabilitation
growing	description	divisions	Holland	spherical
watch	solely	farmer	Russia	Balkan
earth	corresponds	coined	Greece	Finland
studied	appreciation	reviews	adjacent	Bulgaria
anyone	enables	bowling	voluntarily	traction
quarter	profession	compete	massacre	ethnic
apart	capitalism	averaged	devastated	vocalist
crowd	amazing	knights	psychology	coastal
ranges	assembly	muscles	Turkish	television
supplies	abroad	electrical	mobile	conceptual
accompanying	host	China	actress	database
pull	Tchad	Italy	Denmark	Edirne

Çizelge 4.6. Like/Regex aramalarında kullanılan kelimeler.

allows	student	grandmother	Tchad	database
computer	Turkey	psychology	Turkish	Edirne

Her kelimenin Gutenberg veri setinde kaç farklı kitapta, Wikipedia veri setinde ise kaç farklı makale özetinde geçtiği bulunmuş ve bu sorguların ne kadar sürede yapıldığı da elde edilmiştir. Elasticsearch üzerinde bu türden bir sorgulama çok hızlı yapılabildiği için (muhtemelen ilgili doküman sayısı da indeks içinde saklandığından dolayı), sadece doküman/kayıt sayısı değil, ilgili doküman/kayıt içindeki `name` (Gutenberg veri seti için) veya `id` (Wikipedia veri seti için) alanlarındaki bilgilerin de elde edildiği farklı bir yöntem ile de testler yapılmıştır. Testlerde kullanılan iki farklı yöntem kısaca şu şekildedir:

- **Yöntem 1:** MSSQL ve MySQL'de Gutenberg için `COUNT(name)`, Wikipedia için `COUNT(id)` SQL ifadeleri kullanılmıştır. Aynı işlemler için MongoDB'de Python'ın `pymongo` paketinin `count_documents` yöntemi kullanılırken, Elasticsearch'te `Elasticsearch` paketinin `count` yöntemi kullanılmıştır.
- **Yöntem 2:** Aranan kelimenin yer aldığı tüm dokümanlara/kayıtlara ait `name` veya `id` bilgileri bir listeye atılarak, bu liste üzerinde `len` fonksiyonu ile ilgili sayılar elde edilmiştir. Bu yöntem doğal olarak birinci yöntemden daha yavaş sonuç döndürmektedir.

Yapılan tüm testlerde veri tabanları üzerinde ilk önce "set" kelimesi aratılmıştır. Bu işlemin amacı tüm kitapların içeriğinde yer alan bu kelimenin aratılması ile Gutenberg veri setindeki tüm kitaplara ait verinin RAM'e çıkartılmasını sağlamaktır (Wikipedia veri setindeki tüm makale özetlerinde yer alan ortak bir kelime bulunamadığından o veri seti için de aynı kelime kullanılmıştır). Böylece daha sonra yapılacak sorgularda disk erişimi minimuma indirilmiştir (in-Memory olarak çalışmadığı sürece her VT sistemi ilgili veri RAM'de yok ise diskten alıp tampon bölgeye yazma ihtiyacı duyduğu için yapılan ilk sorgu yavaş çalışmaktadır). Tüm testlerdeki farklılıkları özetleyebilmek için, Gutenberg veri seti üzerinde "set" kelimesinin aratılmasında kullanılan sorgular veya ilgili Python yöntemleri aşağıda verilmiştir:

1. MSSQL'de "Yöntem 1 FTS" için 'a', "Yöntem 1 Like/Regex" için 'b' sorguları kullanılmış, Yöntem 2 için `COUNT(name)` yerine `name` yazılmıştır.

a. `SELECT COUNT(name) FROM Books WHERE FREETEXT(content, 'set')`

b. `SELECT COUNT(name) FROM Books WHERE content LIKE ('%set%')`

2. MySQL'in MSSQL'den tek farkı yukarıdaki 'a' sorgusunda `FREETEXT(content, 'set')` yerine `MATCH(content) AGAINST('set' IN BOOLEAN MODE)` kullanılmış olmasıdır. Diğer sorgular aynıdır.
3. MongoDB'de Yöntem 1 için `pymongo` paketindeki `count_documents` (FTS için 'a', Like/Regex için 'b'), Yöntem 2 için ise `find` (FTS için 'c', Like/Regex için 'd') yöntemleri kullanılmıştır:

   a.
   ```
   collection.count_documents({"$text": {"$search": "set"}})
   ```

   b.
   ```
   collection.count_documents({'content': {'$regex': "set"}})
   ```

   c.
   ```
   collection.find({"$text": {"$search": "set"}},
                                          { "_id": 0, "name": 1 })
   ```

   d.
   ```
   collection.find({'content':{"$regex": "set"}},
                                          { "_id": 0, "name": 1 })
   ```

4. Sadece FTS testi yapılan Elasticsearch'de Yöntem 1 için `Elasticsearch` paketindeki `count` (a), Yöntem 2 için ise `search` (b) ifadeleri kullanılmıştır:

   a.
   ```
   es.count(index="gutenberg", body={"query": {"match":
                                             {"content": "set"}}})
   ```

   b.
   ```
   es.search(index="gutenberg", size=10000, scroll='5m',
                    body={"query": {"match": {"content": "set"}},
                           "fields": ["name"], "_source": False})
   ```

4.4.1. İndeksler üzerinde yapılan FTS arama sonuçları

Bir veri tabanı üzerinde FTS testi yapan Python kodunda yukarıda bahsettiğimiz gibi önce "set" kelimesi aranmış, sonra `words` adı verilen liste içinde saklanan 100 adet kelime bir for döngüsü ile arka arkaya sorgulanarak sonuçlar ekranda gösterilmiştir. MSSQL'de Gutenberg veri seti üzerinde FTS testini yapmak için Yöntem 1'e yönelik yazılan kod parçası aşağıdaki gibidir:

```
total = 0
for word in words:
    start = time.perf_counter()
    sql = "SELECT count(name) FROM Books" +
          "WHERE FREETEXT(content, '" + word + "')"
    imlec.execute(sql)
    count = imlec.fetchall()[0][0]
    etime = "%.2f"%(1000*(time.perf_counter()-start))
    print("%s\t %s\t %s" % (word, count, etime))
    total += float(etime)

print("... Total Time: %.2f ms ..." % total)
```

Yöntem 1 ile ilgili işlem tamamlandıktan sonra kodun devamında Yöntem 2 için de benzer bir döngü ile test yapılmıştır. Yukarıdaki kod parçasında SQL ifadesi içindeki `count(name)` yerine `name` kullanılmış ve bunun sonucu olarak artık bir sayı değil de isim listesi elde edileceği için (`liste = imlec.fetchall()`), `print` içinde `count` değişkeni yerine bu listenin eleman sayısını veren `len(liste)` kullanılmıştır.

Aynı işlemleri MySQL üzerinde gerçekleştirmek için `WHERE` fonksiyonundan sonra her iki yöntem için `MATCH(content) AGAINST('" + word + "' IN BOOLEAN MODE)"` ifadeleri kullanılmıştır. Gutenberg veri setlerinde aramalar `content` alanı üzerinden yapılırken Wikipedia veri setlerinde ise `abstract` alanı üzerinden yapılmış ve `name` yerine `id` alanları elde edilmiştir.

MongoDB'deki FTS testlerinde, Yöntem 1 için kullanılan kodun ilgili kısmı aşağıda verilmiştir (aramada bir alan ismi gerekli olmadığından iki veri seti için de kod aynıdır):

```
total = 0
for word in words:
    start = time.perf_counter()
    count = collection.count_documents({"$text":{"$search": word}})
    etime = "%.2f"%(1000*(time.perf_counter()-start))
    print("%s\t %s\t %s" % (word, count, etime))
    total += float(etime)

print("... Total Time: %.2f ms ..." % total)
```

`count_documents` yerine `find` kullanılan Yöntem 2'de sorgu sonucunda bir sayı yerine doküman listesi elde edildiği için MSSQL ve MySQL'dekine benzer şekilde bu listenin eleman sayısı çıktı olarak verilmiştir. Gutenberg veri seti için Yöntem 2'de kullanılan kod aşağıda verilmiştir (diğer veri setinde `{"_id": 0, "name": 1}` yerine `{"_id": 1}` ifadesi kullanılmıştır):

```
liste = list(collection.find({"$text":{"$search": word}},
                             {"_id": 0, "name": 1}))
```

Elasticsearch FTS testlerinde Yöntem 1 için yazılan kodun MongoDB için yazılan koddan tek farkı sorgunun işlendiği satırdır.

```
count = es.count(index="gutenberg",
     body={"query": {"match": {"content": word}}})["count"]
```

Yukarıdaki kod satırında `es` bir Elasticsearch nesnesidir ve test kodunun başında `es = Elasticsearch(timeout=120)` ifadesi ile oluşturulmuştur. Parametre olarak 120 saniye zamanaşımı değeri verilmesinin nedeni Gutenberg veri setinde Yöntem 2 testlerinde bazı kelimeler için varsayılan değer olan 30 saniyenin aşılmasıdır.

Yöntem 1 FTS testlerinde `es` nesnesinin `count` yöntemi kullanılırken, aşağıda kodun ilgili kısmı verilen Yöntem 2 testlerinde ise `search` yöntemi kullanılmıştır:

```
total = 0
for word in words:
    start = time.perf_counter()
    a = es.search(index="gutenberg", size=10000, scroll='5m',
                  body={"query": {"match": {"content": word}},
                        "fields": ["name"], "_source": False})
    sid = a['_scroll_id']
    count = len(a['hits']['hits'])
    while True:
        a = es.scroll(scroll_id = sid, scroll='5m')
        sid = a['_scroll_id']
        count += len(a['hits']['hits'])
        if count % 10000: break
    etime = "%.2f"%(1000*(time.perf_counter()-start))
    print("%s\t %s\t %s" % (word, count, etime))
    total += float(etime)
print("... Total Time: %.2f ms ..." % total)
```

Yukarıdaki kodda görüldüğü gibi `search` yönteminde `count` yöntemine göre daha fazla parametre için değer verilmiştir. *`size`* parametresi sorgudan gelen sonucun en fazla kaç elemanının elde edilebileceğini belirler ve bu değer en fazla 10.000 olabilir. Testlerdeki bazı kelimeler Gutenberg veri setinde 20.000'den fazla kitapta ve Wikipedia veri setinde 100.000'den fazla makale özetinde yer aldığı için bir `while` döngüsü içinde `es.scroll` yöntemi kullanılarak verinin tamamı 10.000'lik parçalar halinde elde edilmiştir. `search` yöntemindeki *`scroll`* parametresi ile bu parçaların ne kadar süreyle muhafaza edilmesi gerektiği belirtilmiştir (aynı parametre `scroll` yönteminde de kullanılmıştır ve verilen '5m' = 5 dakika aslında ihtiyaç duyduğumuz süreden fazladır). *`body`* parametresinde ise `count` yönteminde kullanılan `"query"` kısmı aynı şekli ile kullanılmış, ek olarak `"fields"` kısmında sorgu sonucunda elde edilecek alan (`name`) belirlenmiş, `"_source": False` ile de ilgili dokümanlardaki diğer alanlara ait bilgilerin alınmaması sağlanmıştır.

Wikipedia veri seti ile ilgili test kodundaki temel farklar; aramalar `content` yerine `abstract` üzerinden yapılmış, Yöntem 2 için Elasticsearch tarafından dokümanlara otomatik verilen '`_id`' alanının elde edilmesi yeterli görüldüğünden (benzer bir durum MongoDB için de geçerli idi) `"fields"` ile bir alan belirlenmemiştir.

Tüm VT'ler üzerinde yapılan FTS testlerindeki süre sonuçları Çizelge 4.7'de verilmiştir.

Çizelge 4.7. FTS arama sonuçları (sn).

Veri Tabanı Yönetim Sistemleri	Yöntem 1		Yöntem 2	
	Gutenberg	**Wikipedia**	**Gutenberg**	**Wikipedia**
MSSQL	0,45	8,19	1,16	9,28
MySQL	1,03	1,50	12.845,26	1.168,26
MongoDB	114,32	15,67	114,52	25,59
Elasticsearch	0,22	0,18	4.401,9	322,7

Gutenberg veri seti üzerindeki Yöntem 1 testlerinde en hızlı sonuç veren NoSQL türündeki Elasticsearch olurken, diğer NoSQL türündeki MongoDB 519 kat daha yavaş sonuç döndürerek son sırada yer almıştır. İkinci sırada yer alan MSSQL veri tabanı Elasticsearch'e göre 2 kat daha yavaş sonuç verirken, MySQL'e göre neredeyse 3 kat ve MongoDB'ye göre 254 kat daha hızlı sonuç vermiştir.

Wikipedia veri seti üzerindeki Yöntem 1 testlerinde en hızlı sonuç döndüren yine Elasticsearch olurken, MySQL'e göre 8 kat, MSSQL'e göre 45 kat ve son sırada yer alan MongoDB'ye göre 87 kat daha hızlı olmuştur. Benzer sorgu Gutenberg veri seti üzerinde yapıldığında üçüncü sırada kalan MySQL, daha önce ikinci sırada yer alan MSSQL'den 5 kat daha hızlı sonuç vererek ikinci sıraya yükselmiştir.

Gutenberg veri seti ile ilgili Yöntem 2 testlerinde en hızlı sonuç veren RDBMS türündeki MSSQL veri tabanı olurken en yavaş sonuç veren MySQL olmuştur. MSSQL yaklaşık olarak MongoDB'den 100 kat, Elasticsearch'ten 4 bin kat ve MySQL'den 11 bin kat daha hızlı sonuç vermiştir. MongoDB ikinci sırada yer alarak Elasticsearch'ten 38 kat ve MySQL'den 112 kat daha hızlı sonuç vermiştir.

Wikipedia veri seti ile ilgili Yöntem 2 testlerinde sonuçları en hızlı döndüren MSSQL veri tabanı, MongoDB'ye göre yaklaşık 3 kat, Elasticsearch'e göre 34 kat ve son

sırada yer alan MySQL'e göre 125 kat daha hızlı olmuştur. İkinci sırada yer alan MongoDB'nin sonucu ise Elasticsearch'e göre 12 kat ve MySQL'e göre 45 kat daha hızlı olmuştur.

4.4.2. İndeksli olmayan veriler üzerinde yapılan Like/Regex arama sonuçları

İndeksli olmayan veriler üzerinde yapılan testlerde RDBMS'ler için `LIKE` ve MongoDB için ise `Regex` türü aramalar kullanılmıştır. FTS aramalarındaki gibi bu testte de aynı iki veri seti üzerinde Yöntem 1 ile Yöntem 2 kullanılmıştır. Bu aramalar çok daha yavaş olduğu için bu testte 10 adet kelime aranmıştır (Bkz: Çizelge 4.6). Bu testlerde kullanılan kodlar FTS ile ilgili testlerdekine benzer olup aradaki farklılıklar Bölüm 4.4'te özetlendiği için burada tekrar değinilmemiştir. Yine önceki testlere benzer şekilde Wikipedia veri seti için RDBMS'lerde `name` yerine `id` ve `content` yerine `abstract` kullanılmıştır. MongoDB'de de yine Wikipedia veri seti için `name` olmadan sadece `{"_id": 1}` yazılarak sonuçlar elde edilmiştir.

Tüm VTYS'ler üzerinde yapılan Like/Regex aramalarının süre sonuçları Çizelge 4.8'de verilmiştir.

Çizelge 4.8. Like/Regex arama sonuçları (sn).

Veri Tabanı Yönetim Sistemleri	Yöntem 1		Yöntem 2	
	Gutenberg	Wikipedia	Gutenberg	Wikipedia
MSSQL	6.994,35	428,79	6.990,98	428,59
MySQL	2.443,77	430,16	2.449,21	429,42
MongoDB	190,74	64,76	173,11	66,15

Çizelge 4.8'deki sonuçlara baktığımızda, yapılan tüm testlerde en hızlı sonuç veren NoSQL türündeki MongoDB olduğunu söyleyebiliriz. MongoDB, Gutenberg - Yöntem 1 testlerinde en yavaş sonuç döndürerek son sırada yer alan MSSQL'den 37 kat, ikinci sırada yer alan MySQL'den ise 13 kat daha hızlı sonuç vermiştir.

Wikipedia - Yöntem 1 testlerinde en hızlı sonuç döndüren yine MongoDB olurken, RDBMS sistemler neredeyse aynı değerleri üreterek MongoDB'den yaklaşık 7 kat daha yavaş sonuç vermişlerdir. Gutenberg - Yöntem 1 üzerinde son sırada yer alan MSSQL, Wikipedia üzerinde MySQL'den biraz daha hızlı sonuç vermiştir.

Like/Regex türü aramaların yavaş olması sebebiyle, Yöntem 2'de Yöntem 1'e göre daha fazla veri gönderilmesine rağmen, iletim süresi arama süresinin yanında çok küçük olduğu için, her iki yöntemin sonuçları arasında fazla fark olmamıştır.

BÖLÜM 5

SONUÇLAR

Bu tez çalışmasının genel amacı, farklı metin indeksleme tekniklerini kullanan farklı veri tabanı sistemleri üzerindeki Tam Metin Arama yaklaşımlarını incelemek ve performanslarını kıyaslamak olmuştur. İndeksli olmayan metin verisi üzerinde 'Like' veya 'Regex' gibi yöntemler ile normal metin arama performansının ölçülmesi de çalışmanın yan amaçlarından biri olmuştur.

Çalışma kapsamında seçilen veri tabanı yönetim sistemlerine yüklenen iki farklı veri seti üzerinde tam metin araması yapılmıştır. Bu iki veri setinden biri daha az sayıda ama büyük boyutlu metinleri (kitaplar) içerirken, diğeri daha çok sayıda ama küçük boyutlu metinleri (makale özetleri) içermektedir.

Süre ölçümlerinde adil olunması açısından tamamı Python kodları ile yapılan testlerde elde edilen sonuçlara göre, verilen kelimeyi içeren kitapların isimlerini getirme sorgusu yürütüldüğünde en yüksek performans gösteren RDBMS kategorisindeki MSSQL olurken, en düşük hız yine aynı kategoride olan MySQL'de kaydedilmiştir. Bu sorgu testinde ikinci sırayı MongoDB, üçüncü sırayı ise Elasticsearch almıştır. Verilen kelimeyi içeren kitapların sayısını getirme sorgusunda ise NoSQL kategorisinde yer alan Elasticsearch arama motoru en hızlı olurken, aynı kategoriden olan MongoDB veri tabanı en yavaş olmuştur. Bu sorgu testinde ise ikinci sırayı MSSQL, üçüncü sırayı MySQL almıştır.

Verilen kelimeyi içeren *makale özetlerinin 'id' numaralarını* getirme sorgusunda en yüksek performans gösteren yine MSSQL olurken, MySQL de yine en yavaş olmuştur. Kitaplar ile ilgili sorgu testinde olduğu gibi ikinci sırada yine MongoDB ve üçüncü sırada

yine Elasticsearch yer almıştır. Benzer şekilde, *verilen kelimeyi içeren makale özetlerinin sayısını* getirme sorgusunda da yine Elasticsearch en yüksek ve MongoDB en yavaş performansa sahip olmuş, fakat bu defa ikinci ve üçüncü sıra yer değiştirmiştir. Kitaplar ile ilgili testte MySQL'e göre 2 kat daha hızlı sonuç veren MSSQL, makale özetleri söz konusu olduğunda 5 kat yavaş kalmıştır.

Elasticsearch'ün yer almadığı Regex/Like testlerindeki sonuçlara göre, Regex kullanan MongoDB her durumda Like türü arama yapan rakiplerine karşı üstün performans göstermiştir. Kitaplar üzerinde MySQL ikinci MSSQL ise üçüncü olurken, makale özetleri ile ilgili sorgularda her iki RDBMS sistemi de yakın sonuçlar vermiştir.

Yapılan test sonuçlarından yola çıkarak, metin indeksi üzerinde yapılan aramalarda sadece belirli bir kelimeyi içeren belgelerin sayısı isteniyorsa Elasticsearch arama motorunun tercih edilmesi mantıklıdır. Fakat belgeler ile ilgili bir alanın listesi elde edilecek ise MSSQL veri tabanının kullanılmasının performansı çok daha iyi olacaktır. İndeks kullanmayan metin içinde metin arama sorgularında ise MongoDB veri tabanının çok iyi performans verdiği görülmüştür. MySQL ise herhangi bir durumda en iyi olamamıştır. Teste dahil edilen tüm veri tabanları ile ilgili genel değerlendirmenin özeti Çizelge 5.1'de verilmiştir.

Çizelge 5.1. Arama türüne göre en iyi veri tabanları.

Veri Tabanı Yönetim Sistemleri	**Metin İndeksleri Üzerinde Arama**		**Normal Arama**
	İlgili belgelere ait bir alan (name / id) isteniyorsa	**İlgili belge sayısı (count) isteniyorsa**	**Like/Regex sorgularında**
MSSQL	En hızlı		En yavaş
MySQL	En yavaş		
MongoDB		En yavaş	En hızlı
Elasticsearch		En hızlı	

KAYNAKLAR

Aladily, A. (2015). *The performance wise comparison of the most widely used noSQL databases*. Yüksek Lisans Tezi. Kadir Has Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

Ayan, Z. (2017, 3 Ağustos). Apache Solr: Bir Arama Motorundan Fazlası? 7 Ekim 2020 tarihinde http://devnot.com/2017/apache-solr-bir-arama-motorundan-fazlasi/ adresinden erişildi.

Balıkçı, E. (2020, 10 Temmuz). Elasticsearch Nedir ve Nasıl Kullanılır? 20 Ocak 2022 tarihinde https://www.datasciencearth.com/Elasticsearch-nedir-ve-neden-kullanilir/ adresinden erişildi.

Bhamra, K. (2017). *A Comparative Analysis of MongoDB and Cassandra*.Yüksek Lisans Tezi. University of Bergen / Department of Informatics, Norway.

Bolat N. A. (2019). *NOSQL ve RDBSM yapıların performans karşılaştırması*. Yüksek Lisans Tezi. Beykent Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 13*(6), 377-387.

Dayıbaşı, E. (2016, 18 Mayıs). CAP Teorimi Nedir? Medium.com / Cloud And Servers. 06 Nisan 2020 tarihinde https://medium.com/cloud-and-servers/cap-teoremi-nedir-8673e71c10fc adresinden erişildi.

Delen, F. (2021, 17 Ocak). Elasticsearch Nedir, Nasıl Kullanılır? 20 Ocak 2022 tarihinde https://fatmadelenn.medium.com/elasticseacrh-nedir-nas%C4%B1l-kullanılır-d99c28b3e779 adresinden erişildi.

Doğan, K. (2019, 3 Şubat). Elasticsearch Nedir? 20 Ocak 2022 tarihinde https://medium.com/@kdrcandogan/elasticsearch-nedir-45d237c29b26 adresinden erişildi.

Dumanlı, B. (2019). *En çok kullanılan ilişkisel ve NoSQL veri tabanı yönetim sistemlerinin performans karşılaştırması*. Yüksek Lisans Tezi. Trakya Üniversitesi / Fen Bilimleri Enstitüsü, Edirne.

Elias, J. (2017, 25 Ağustos). Hand on Full-Text Search in SQL Server. SQLShack. 20 Ocak 2022 tarihinde https://www.sqlshack.com/hands-full-text-search-sql-server/ adresinden erişildi.

Gökalp, G. (2015, 22 Ekim). Lucene.Net Search Engine Kütüphanesi Kullanımı. 21 Ocak 2022 tarihinde https://www.gokhan-gokalp.com/lucene-net-search-engine-kutuphanesi-kullanimi/ adresinden erişildi.

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

Hammood, A. H. (2016). *NoSQL veritabanı sistemlerinin karşılaştırılması: MongoDB, apache HBase, and apache Cassandra üzerine bir çalışma*. Yüksek Lisans Tezi. Çankaya Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara.

IBM Cloud Education. (2019, 14 Kasım). CAP Teorimi Nedir? 06 Nisan 2020 tarihinde https://www.ibm.com/tr-tr/cloud/learn/cap-theorem adresinden erişildi.

Kılıç, U., Karabey, I. (2016, October). Comparison of Solr and Elasticsearch Among Popular Full Text Search Engines and Their Security Analysis. In *International Conference on Computer Science and Engineering* (pp. 812-818*)*, Tekirdağ / Türkiye.

MentorMate. (2017, 1 Şubat). Elasticsearch: How to Add Full-Text Search to Your Database. 16 Ocak 2022 tarihinde https://mentormate.medium.com/Elasticsearch-how-to-add-full-text-search-to-your-database-ee2f3ea4d3f3 adresinden erişildi.

Mesut, A., Öztürk, E. (2021). A method to improve full-text search performance of MongoDB. *Pamukkale University Journal of Engineering Sciences*, https://jag.journalagent.com/pajes/pdfs/PAJES-89590-RESEARCH_ARTICLE-MESUT.pdf

Minack, E., Siberski, W., & Nejdl, W. (2009, Mayıs). Benchmarking fulltext search performance of RDF stores. In *European Semantic Web Conference* (pp. 81-95). Springer, Berlin, Heidelberg.

MongoDB. (2022, 19 Mayıs). Wikipedia. The free encyclopedia içinde. 21 Mayıs 2021 tarihinde https://en.wikipedia.org/wiki/MongoDB adresinden erişildi.

MySQL. (2022, 10 Mayıs). Wikipedia, The Free Encyclopedia içinde. 18 Mayıs 2022 tarihinde https://en.wikipedia.org/wiki/MySQL adresinden erişildi.

NoSQL. (2022, 13 Mayıs). Wikipedia, The Free Encyclopedia içinde. 21 Mayıs 2022 tarihinde https://en.wikipedia.org/wiki/NoSQL adresinden erişildi.

Oracle Türkiye. (2021). İlişkisel Veri Tabanı Nedir? 16 Ocak 2021 tarihinde https://www.oracle.com/tr/database/what-is-a-relational-database/ adresinden erişildi.

DB-Engines. (2022, 1 Mayıs). DB-Engines / Knowledge Base of Relational and NoSQL Database Management Systems içinde. 7 Mayıs 2022 tarihinde https://db-engines.com/en/ranking adresinden erişildi.

Seçgin C. (2018). *İlişkisel ve NoSQL Veri Tabanı Sistemlerinin Performans Karşılaştırması*. Yüksek Lisans Tezi. İstanbul Aydin Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

Shwaysh, M. M. (2018). *Security and performance comparison of NoSQL database systems*.Yüksek Lisans Tezi. Çankaya Üniversitesi / Fen Bilimleri Enstitüsü, Ankara.

Singh, K. (2015). *Survey of NoSQL Database Engines for Big Data*. Yüksek Lisans Tezi, Aalto University, School of Science, Espoo / Finland.

Sumukh, B. & Nandita, B. J. (2020). A Full-Text-Based Search Algorithm vs Elasticsearch, *Studies in Indian Place Names, 40*(7), 2168-2171.

Swaminathan, S. N. (2015). *Quantıtatıve analysıs of scalable nosql databases*. Yüksek Lisans Tezi. The University of Texas at Arlington, ABD.

Taha, İ. A. (2017). *A comprehensive comparison of NOSQL and relational database management systems,* Yüksek Lisans Tezi, Çankaya Üniversitesi / Fen Bilimleri Enstitüsü, Ankara.

Trelle, T. (2013, 1 Temmuz). *MongoDB Text Search Explained*. Codecentric. 16 Ocak 2022 tarihinde https://blog.codecentric.de/en/2013/01/text-search-mongodb-stemming/ adresinden erişildi.

Vileikis, L. (2020, 13 Kasım). *Full-Text Searches in MySQL: The Good, the Bad and the Ugly*. 20 Ocak 2022 tarihinde https://severalnines.com/database-blog/full-text-searches-mysql-good-bad-and-ugly adresinden erişildi.

w3resource. (2022, 4 Nisan). MySQL Full text search. 20 Mayıs 2022 tarihinde https://www.w3resource.com/mysql/mysql-full-text-search-functions.php adresinden erişildi.

Walker, J. (2021, 16 Mart). How to Use Full-Text Searches in MySQL. Cloud Savvy IT. 20 Ocak 2022 tarihinde https://www.cloudsavvyit.com/10172/how-to-use-full-text-searches-in-mysql/ adresinden erişildi.

Yang, C. T., Kristiani, E., Wang, Y. T., Min, G., Lai, C. H., & Jiang, W. J. (2020). On construction of a network log management system using ELK Stack with Ceph. *The Journal of Supercomputing*, *76*(8), 6344-6360.

EKLER

MSSQL Server FTS Arama Sonuçları

Kelime	Gutenberg			Wikipedia		
	Kelime Sayısı	Yöntem 1 Arama Süresi (ms)	Yöntem 2 Arama Süresi (ms)	Kelime Sayısı	Yöntem 1 Arama Süresi (ms)	Yöntem 2 Arama Süresi (ms)
allows	21405	7,24	23,4	67102	53,90	64,32
computer	21385	6,38	24,25	39231	66,68	64,92
original	21316	5,85	24,57	102022	620,09	635,86
turns	21167	8,8	24,11	56005	43,56	59,84
staff	21034	6,32	24,68	31948	22,75	42,87
feels	20997	8,83	23,68	18232	16,23	21,26
fired	20517	7,51	23,42	40186	29,52	42,60
attachment	7296	3,58	9	1796	4,58	7,75
growing	20873	8,88	34,58	72666	49,14	75,25
watch	20036	7,5	22,57	12596	10,11	31,43
earth	19821	6,25	22,18	25463	23,13	28,36
studied	19283	7,01	21,6	124109	600,23	614,76
anyone	19028	5,22	20,85	4782	10,32	18,72
quarter	18701	6,35	21,1	16383	12,29	34,70
apart	17034	5,37	18,8	11376	10,34	24,52
crowd	18450	6,63	19,35	7421	15,76	17,24
ranges	15562	5,68	16,78	81973	596,93	610,40
supplies	18004	6,5	19,73	25484	19,47	29,89
accompanying	17479	6,7	19	16334	13,85	35,48
pull	17470	6,4	19,46	7300	15,18	17,86
proposes	16628	6,51	18,38	22781	17,83	23,64
seasons	17101	6,26	18,81	191391	605,14	626,29
handles	17071	6,43	18,46	15404	13,71	43,25
grand	16608	5,51	18,57	47338	30,48	43,77
England	16591	5,57	17,76	134378	598,13	616,05
conclusion	16120	5,47	17,24	6484	13,72	22,13
retiring	15928	5,78	17,42	77601	47,61	67,69
judgment	16726	5,82	18,53	4093	9,84	20,32
description	15839	5,49	17,86	10099	10,08	22,10
solely	8516	3,96	9,86	5220	11,75	12,29
corresponds	12137	4,86	13,62	10921	11,78	24,11

appreciation	8752	4,17	10,32	2074	5,51	6,81
enables	15276	5,77	16,64	14840	13,38	31,91
profession	12774	5,25	14,15	7203	14,93	17,61
capitalism	360	2,89	3,71	1348	3,78	4,45
amazing	12156	4,77	13,01	2909	6,95	8,53
assembly	8829	4,14	10,29	53195	28,86	45,65
abroad	14275	4,54	15,51	9315	19,25	22,95
host	14078	5,71	15,64	68842	44,33	59,19
Tchad	55	0,6	1,9	87	0,80	6,69
characteristics	13572	5,43	14,68	15303	15,29	34,11
religious	13283	4,77	14,91	28968	23,13	30,90
artist	11561	4,86	12,95	104579	595,07	613,17
ocean	13079	5,07	29,85	26253	19,77	32,01
farming	12840	5,16	14,17	33246	22,38	33,49
achievement	8283	4,51	9,57	15079	12,05	33,63
overlooking	12220	4,93	13,74	8103	17,47	19,98
student	11981	5,15	12,83	77399	592,80	605,19
divisions	11780	4,78	12,96	107164	595,04	611,69
farmer	11235	5,2	12,95	17273	12,66	24,44
coined	10454	4,98	11,32	10895	11,82	24,97
reviews	10180	5	11,31	43454	26,11	44,35
bowling	10102	4,45	11,85	21230	14,18	24,36
compete	4247	3,19	5,36	108032	599,14	613,56
averaged	10158	4,63	11,24	30000	22,06	34,57
knights	9567	4,59	10,9	20022	13,37	21,24
muscles	9790	4,6	10,75	5016	11,62	13,07
electrical	2510	2,75	4,16	9684	9,07	22,07
China	9532	4,64	10,23	62545	33,92	53,62
Italy	9308	4,07	10,48	48908	33,24	54,39
Germany	8794	3,99	10,25	96330	597,78	611,26
Spain	8289	4,16	9,24	41676	25,99	41,73
eventually	7471	3,55	8,79	40849	32,80	47,07
champion	7881	4,38	8,82	64870	37,21	57,83
Mexico	7410	3,81	8,52	44107	36,13	52,19
Egypt	7318	3,92	8,14	12730	10,17	28,62
Turkey	7014	3,7	8,12	26099	18,11	25,07
grandmother	6762	3,68	7,86	2565	6,44	8,35
Holland	6075	3,81	7,01	6509	14,28	15,94
Russia	6242	3,54	7,2	39149	23,66	34,90
Greece	5784	3,81	6,62	18593	14,09	20,19
adjacent	5509	3,31	6,5	20303	17,91	22,81
voluntarily	5437	3,27	6,33	1240	3,52	5,06
massacre	5320	3,72	6,46	5358	12,25	12,79
devastated	3437	3,12	4,54	2663	6,65	7,29
psychology	2769	3,25	3,56	7622	15,72	17,58

Turkish	4294	3,31	5,11	18369	12,87	20,11
mobile	2675	2,86	3,61	14479	11,79	31,27
actress	3298	3,35	4,44	44441	30,55	50,80
Denmark	3018	2,94	3,99	15702	10,76	38,73
Norway	2873	3,27	4,31	26598	17,03	25,63
Sweden	2830	3,07	3,81	28021	18,02	27,24
anthem	2080	2,81	3,27	3589	8,74	10,46
broadcast	1906	3,11	3,49	61706	42,76	58,01
Jerome	1837	2,73	3,13	3030	7,21	10,92
Hannah	1589	3,15	3,47	2347	5,95	7,08
phone	1381	2,78	3,18	8357	17,46	18,87
rehabilitation	625	2,41	2,84	3183	7,51	8,81
spherical	1003	2,73	3,03	1542	3,95	4,78
Balkan	848	2,7	2,95	4106	9,46	10,43
Finland	815	2,72	2,71	15538	11,20	32,99
Bulgaria	704	2,49	2,83	10893	8,89	23,00
traction	555	2,51	2,68	1308	4,23	4,74
ethnic	378	2,82	2,9	14005	12,92	29,54
vocalist	412	2,77	2,79	14786	13,10	31,87
coastal	347	2,35	2,41	16433	13,65	34,66
television	205	2,74	3,13	127671	596,49	611,57
conceptual	118	2,32	2,52	2201	9,53	12,81
database	64	2,9	2,64	7900	17,02	17,91
Edirne	18	0,51	0,47	219	1,08	2,39
Toplam Süre		**449,63**	**1160,67**		**8.186,99**	**9.281,57**

MySQL FTS Arama Sonuçları

Kelime	**Gutenberg**			**Wikipedia**		
	Kelime Sayısı	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**	**Kelime Sayısı**	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**
allows	6159	13,44	108.980,73	18059	18,98	12.035,32
computer	21385	36,78	173.859,00	34805	28,45	22.959,00
original	21316	33,43	311.961,90	101225	75,05	40.962,72
turns	15275	18,67	258.766,81	5562	12,94	4.876,84
staff	21008	22,59	295.718,70	29788	60,58	18.776,35
feels	12810	17,33	213.640,85	1526	5,72	2.043,36
fired	11174	12,17	195.542,94	6750	10,45	5.990,28
attachment	6847	8,63	138.634,77	1502	3,88	2.090,05
growing	17977	24,6	267.597,78	22522	21,09	14.575,81
watch	18795	32,48	294.232,72	6418	6,03	6.058,20
earth	19814	33,87	317.450,42	25317	16,94	16.853,29
studied	13632	13,76	231.898,83	47309	32,20	25.011,63
anyone	19038	27,1	271.530,48	4837	4,31	4.672,17
quarter	17049	23,04	296.580,84	12607	10,86	9.800,40

apart	17035	16,81	275.318,97	11376	7,92	7.940,40
crowd	16208	22,18	278.436,00	4637	4,56	4.648,33
ranges	4237	5,2	88.746,92	8648	5,90	7.510,54
supplies	9733	9,72	181.230,87	6514	5,53	6.045,78
accompanying	7676	6,6	152.960,09	4302	4,12	4.136,89
pull	13546	15,76	229.302,16	2454	2,39	3.104,41
proposes	3731	4,57	77.597,90	1013	1,88	1.507,15
seasons	8493	9,21	157.996,73	42111	28,20	25.404,17
handles	3857	6,06	75.984,53	1977	2,92	2.587,92
grand	16594	19,81	277.433,86	47109	29,89	28.483,89
England	16593	40,29	289.633,72	134378	92,14	47.935,32
conclusion	15364	20,08	269.902,84	5553	4,40	4.792,24
retiring	6999	6,78	142.267,19	9242	6,52	6.902,05
judgment	16056	24,46	280.361,79	2439	3,56	2.677,85
description	15228	20,01	252.847,43	7686	6,21	6.226,54
solely	8515	8,7	161.707,12	5220	4,53	4.347,14
corresponds	3161	3,65	72.964,19	2435	2,49	2.804,10
appreciation	8687	9,24	155.298,26	2040	2,89	2.788,03
enables	4902	5,52	101.493,87	3591	3,29	3.656,07
profession	11948	13,88	221.554,85	6100	4,64	5.246,52
capitalism	360	2,86	9.128,31	1348	2,09	1.944,40
amazing	7016	8,88	128.714,94	2641	2,77	3.399,34
assembly	8297	13,68	167.001,57	52055	38,55	29.554,45
abroad	14275	16,33	254.489,54	9315	7,21	7.502,86
host	12959	15,96	230.404,60	30236	20,60	18.529,88
Tchad	55	1,92	2.261,65	87	1,51	183,35
characteristics	8126	9,54	148.874,77	8435	6,75	6.482,99
religious	13282	17,97	239.485,31	28968	20,27	17.843,00
artist	10127	11,74	157.051,15	72174	53,78	38.073,44
ocean	12767	19,17	187.204,41	24156	22,72	16.659,05
farming	4035	6,14	66.691,98	7785	6,37	6.737,22
achievement	6072	6,28	116.353,16	8929	6,64	6.917,93
overlooking	4947	6,1	71.943,93	4597	4,43	4.081,03
student	10016	10,43	150.137,58	31675	21,82	21.323,88
divisions	6503	7,71	136.000,07	11993	8,91	8.861,73
farmer	9416	10,97	171.894,19	9825	6,22	7.839,69
coined	1906	3,89	47.512,26	5762	4,70	4.665,62
reviews	2659	3,54	59.282,51	18529	12,54	12.945,75
bowling	1981	3,24	42.325,26	6014	5,95	6.188,72
compete	3148	3,61	60.424,71	18060	12,43	12.140,67
averaged	1035	2,27	21.186,61	2293	2,67	3.030,55
knights	5453	10,02	110.669,07	5158	4,86	5.774,04
muscles	7521	10,82	139.107,07	1724	2,66	2.260,66
electrical	2510	4,67	47.380,55	9684	6,40	8.209,62
China	9521	12,82	166.606,82	62540	43,77	33.276,27

Italy	9308	13,85	175.175,91	48908	35,32	25.766,37
Germany	8791	14,04	166.291,50	96325	67,42	36.634,13
Spain	8290	14,49	170.907,89	41676	35,85	22.670,74
eventually	7471	8,61	135.210,60	40849	27,11	20.136,65
champion	6575	6,76	127.599,88	41273	29,20	24.626,70
Mexico	7410	11,36	111.351,96	44863	31,50	24.924,95
Egypt	7318	12,04	149.014,53	12731	8,83	10.750,26
Turkey	6431	8,06	129.385,52	25938	17,33	15.549,17
grandmother	6246	8,55	120.016,19	2477	2,97	3.201,76
Holland	6075	8,65	124.973,32	6509	5,78	6.263,93
Russia	6242	10,89	127.581,84	39149	27,99	21.578,26
Greece	5785	8,23	117.217,89	18593	13,46	13.881,04
adjacent	5509	5,79	115.720,50	20303	14,05	12.728,88
voluntarily	5437	7,31	113.819,77	1240	2,38	1.913,45
massacre	4266	5,46	91.780,86	4466	3,93	5.397,16
devastated	1795	2,72	42.080,67	1223	1,95	1.716,64
psychology	2766	4,36	51.788,82	7613	5,80	8.312,08
Turkish	4294	5,84	93.219,04	18370	13,79	15.380,91
mobile	2658	3,97	60.549,21	14398	11,12	11.950,92
actress	2758	3,23	58.662,30	43332	28,62	25.031,84
Denmark	3017	4,52	63.382,48	15702	12,34	12.274,48
Norway	2873	5,5	61.853,47	26599	17,26	17.045,90
Sweden	2829	4,98	64.005,61	28021	17,67	18.358,17
anthem	1631	2,67	36.055,84	3137	4,15	3.752,79
broadcast	1836	3,34	38.086,63	36449	24,05	22.185,84
Jerome	1837	3,2	41.607,25	3595	4,15	3.765,23
Hannah	1588	4,5	38.066,72	2348	2,95	2.926,65
phone	1249	4,9	21.817,25	6506	6,08	6.485,69
rehabilitation	612	1,98	15.421,29	3180	3,65	4.220,06
spherical	1003	1,89	27.019,03	1542	2,55	2.001,25
Balkan	644	1,93	15.329,52	2561	3,90	3.070,14
Finland	815	1,99	21.883,17	15538	12,12	11.660,82
Bulgaria	704	2,67	17.496,29	10894	8,27	8.185,55
traction	549	1,51	12.103,04	1304	2,28	1.773,85
ethnic	374	1,65	10.460,75	13981	9,47	10.280,90
vocalist	279	1,21	6.858,14	13552	9,02	9.109,77
coastal	347	1,94	8.369,41	16433	11,76	11.186,82
television	204	1,51	4.761,59	127646	100,53	52.193,99
conceptual	118	1,11	3.100,37	2201	2,80	2.587,61
database	61	1,96	2.362,36	6693	5,40	6.542,87
Edirne	18	0,93	1.306,00	219	1,12	360,57
Toplam Süre		**1027,08**	**12.845.262,49**		**1.504,00**	**1.168.261,69**

MongoDB FTS Arama Sonuçları

Kelime	Gutenberg			Wikipedia		
	Kelime Sayısı	Yöntem 1 Arama Süresi (ms)	Yöntem 2 Arama Süresi (ms)	Kelime Sayısı	Yöntem 1 Arama Süresi (ms)	Yöntem 2 Arama Süresi (ms)
allows	21405	2.212,97	2.118,88	67915	310,60	456,66
computer	21388	2.062,77	2.235,22	46433	197,01	347,42
original	21331	2.121,37	2.164,80	249229	1.136,03	1.849,64
turns	21187	2.184,03	2.055,86	56038	237,80	468,64
staff	21029	2.066,69	1.953,43	30146	131,32	248,06
feels	20837	2.114,77	2.107,52	13309	57,65	93,31
fired	20518	2.226,57	2.406,54	40231	196,70	282,32
attachment	20319	2.249,68	3.035,28	14888	84,62	104,93
growing	20260	2.638,06	2.131,99	41455	191,21	284,53
watch	20161	2.203,71	1.977,43	12741	77,27	93,15
earth	19958	2.135,15	2.255,60	25884	128,63	194,55
studied	19300	2.201,13	2.103,43	124279	545,84	916,05
anyone	19038	2.209,53	2.122,42	4856	22,82	67,40
quarter	18829	1.984,40	1.898,51	21111	94,00	151,32
apart	18744	1.930,14	2.013,34	19010	88,37	143,40
crowd	18450	1.924,70	1.818,89	7421	54,47	52,01
ranges	18148	1.979,84	1.919,37	82450	381,31	563,94
supplies	18021	1.891,53	1.923,48	25490	108,35	218,62
accompanying	17919	1.936,41	1.956,51	17755	76,65	124,77
pull	17478	1.790,47	1.819,39	7308	39,74	63,13
proposes	17245	1.751,76	2.266,05	28583	125,24	187,30
seasons	17214	1.726,31	1.934,13	195552	835,11	1.419,26
handles	17072	1.656,78	1.731,96	15440	97,68	182,45
grand	16984	1.728,30	1.771,49	54886	238,99	370,66
England	16630	1.690,46	1.789,08	134454	573,67	976,63
conclusion	16509	1.814,54	1.774,41	7060	51,51	89,60
retiring	16305	1.666,47	1.719,86	93556	397,63	663,30
judgment	16249	1.813,79	1.793,38	2943	15,59	61,94
description	16153	1.711,97	1.745,42	11930	53,23	86,41
solely	15826	1.797,53	1.717,01	15256	74,81	118,86
corresponds	15505	1.887,77	1.690,96	19427	92,17	142,07
appreciation	15392	1.673,20	1.663,49	4724	22,59	34,77
enables	15276	1.789,04	1.614,77	14963	66,12	99,33
profession	15204	1.791,18	1.695,82	8038	52,81	70,56
capitalism	14742	1.571,93	1.729,35	99005	383,58	647,00
amazing	14775	1.407,76	1.730,38	3166	14,87	36,47
assembly	14510	1.668,17	1.637,70	58279	247,24	399,65
abroad	14285	1.600,17	1.516,81	9315	42,65	77,77
host	14146	1.603,04	1.516,54	68911	297,15	476,34
characteristics	13727	1.583,81	1.522,94	15844	69,53	112,19
religious	13629	1.468,06	1.494,03	29261	135,88	192,08

artist	13136	1.367,50	1.443,86	114587	487,00	822,49
ocean	13109	1.483,64	1.403,20	27823	117,81	209,52
farming	12875	1.392,02	1.413,24	33261	144,71	241,48
achievement	12561	1.347,57	1.381,07	49526	210,18	354,75
overlooking	12250	1.338,54	1.480,76	8105	44,44	62,78
student	11989	1.269,97	1.342,82	77410	333,34	544,91
divisions	11848	1.289,37	1.310,74	107563	471,73	774,48
farmer	11242	1.204,31	1.213,72	17274	84,01	127,76
coined	10486	1.116,17	1.203,16	10902	56,36	82,21
reviews	10387	1.243,37	1.128,90	46028	213,84	354,95
bowling	10391	1.167,34	987,96	21698	94,35	147,30
compete	10182	1.211,33	1.208,30	110482	455,40	776,16
averaged	10167	1.076,92	1.120,97	30013	144,17	216,82
knights	9796	1.059,73	971,07	20110	87,54	150,89
muscles	9790	1.071,99	975,17	5020	22,51	37,48
electrical	9631	1.013,89	1.018,50	33072	143,74	216,74
China	9533	986,66	981,06	62549	268,93	449,89
Italy	9345	1.059,39	1.072,04	48928	207,38	400,92
Germany	8839	1.006,82	999,70	96361	409,62	665,72
Spain	8340	981,06	915,17	41693	178,37	314,49
eventually	8118	898,78	943,89	45249	191,94	319,00
champion	7885	867,08	893,87	64874	284,21	457,90
Mexico	7413	516,40	492,18	44863	190,13	324,36
Egypt	7338	758,37	785,54	12739	63,83	111,05
Turkey	7015	694,48	751,88	26100	114,79	173,05
grandmother	6806	706,16	609,45	2567	12,51	22,17
Holland	6404	774,21	759,26	6865	31,20	47,22
Russia	6303	711,83	669,63	39156	166,85	262,68
Greece	5788	686,00	624,97	18594	81,77	127,86
adjacent	5525	679,88	484,92	20376	90,61	146,72
voluntarily	5440	601,59	662,98	1240	9,39	13,04
massacre	5327	615,01	661,95	5361	25,66	37,69
devastated	4998	571,47	576,01	3185	16,16	23,68
psychology	4563	498,21	513,93	12352	55,61	82,72
Turkish	4297	409,52	473,80	18370	80,43	139,30
mobile	3684	373,30	434,83	18624	90,57	123,01
actress	3298	335,53	338,53	44442	206,90	301,33
Denmark	3047	296,15	337,13	15707	78,06	108,09
Norway	2877	289,28	225,53	26606	122,65	181,37
Sweden	2832	278,20	213,44	28024	119,85	204,65
anthem	2094	225,86	182,17	3714	17,21	43,67
broadcast	1909	204,59	221,74	67047	289,44	465,41
Jerome	1885	222,58	198,82	3043	15,83	30,18
Hannah	1591	168,91	144,99	2354	11,48	18,52
phone	1374	118,88	96,14	8359	38,42	57,18

rehabilitation	1164	122,11	91,56	4300	21,37	37,26
spherical	1071	94,43	81,45	1584	8,56	13,43
Balkan	855	72,18	96,10	4127	18,76	29,43
Finland	842	107,98	76,64	15548	68,51	107,65
Bulgaria	706	55,68	37,77	10895	53,72	72,34
traction	557	56,49	50,29	1310	10,15	17,20
ethnic	484	41,51	35,05	17769	86,21	117,71
vocalist	413	48,13	42,97	14789	72,41	108,31
coastal	348	26,54	23,87	16436	82,03	111,43
television	209	4,91	7,98	130084	564,14	920,53
conceptual	156	15,05	12,96	3200	15,44	42,49
database	64	3,29	4,13	7902	35,10	53,81
Tchad	55	10,43	11,23	87	1,45	11,25
Edirne	18	1,00	1,43	219	2,09	4,66
Toplam Süre		**114.315,55**	**114.515,89**		**15.669,31**	**25.589,58**

Elasticsearch FTS Arama Sonuçları

Kelime	**Gutenberg**			**Wikipedia**		
	Kelime Sayısı	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**	**Kelime Sayısı**	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**
allows	6159	2,22	42.536,73	18056	2,53	3.290,04
computer	21385	1,79	115.920,66	34549	2,36	6.238,26
original	21316	2,42	112.946,77	101076	5,65	17.915,84
turns	15275	2,09	89.017,32	5548	1,59	987,56
staff	21008	1,58	110.075,23	29441	1,63	5.083,80
feels	12810	1,46	76.445,17	1523	2,21	282,43
fired	11174	1,56	68.586,95	6733	1,69	1.174,50
attachment	6845	1,50	47.612,78	1498	1,66	275,69
growing	17978	1,49	99.459,96	22477	1,37	3.876,70
watch	18773	2,71	101.243,32	6375	1,78	1.113,63
earth	19786	2,55	104.805,33	22986	1,82	4.065,92
studied	13633	2,27	80.053,27	47276	2,31	8.099,95
anyone	19028	2,84	99.713,24	4768	1,73	879,35
quarter	16994	2,02	95.161,00	12563	1,60	2.137,32
apart	17035	1,91	95.709,83	10940	1,77	1.935,82
crowd	16202	2,20	91.395,43	4593	1,62	812,66
ranges	4237	1,93	28.087,71	8593	1,62	1.484,78
supplies	9733	1,98	60.034,95	6475	1,70	1.150,63
accompanying	7677	2,22	50.506,98	4301	1,76	764,14
pull	13481	1,98	78.146,85	2451	1,75	443,44
proposes	3731	2,05	27.609,62	1011	1,67	184,17
seasons	8495	2,39	54.100,38	41791	1,63	7.071,50
handles	3857	2,06	25.304,37	1971	1,54	363,46
grand	16592	2,69	93.302,37	46913	1,55	7.955,61

England	16551	2,76	91.531,45	131574	2,04	22.472,47
conclusion	15365	2,14	89.387,63	5525	1,83	962,70
retiring	6999	2,24	47.974,63	9185	1,83	1.580,38
judgment	16053	2,67	90.960,49	2425	1,99	446,34
description	15228	2,00	88.632,06	7648	3,37	1.356,19
solely	8516	2,08	56.257,00	5218	2,91	924,47
corresponds	3161	2,49	22.104,93	2435	2,19	464,51
appreciation	8687	2,24	53.622,49	2033	1,63	363,15
enables	4902	2,73	32.565,22	3590	1,80	630,93
profession	11948	2,39	73.431,11	5914	1,76	1.020,47
capitalism	362	1,96	2.591,18	1332	1,62	251,02
amazing	7015	2,58	43.680,30	2638	1,76	472,98
assembly	8281	2,30	54.204,62	51784	1,69	8.463,79
abroad	14276	2,18	83.007,08	9113	1,59	1.580,31
host	12857	2,33	78.220,68	30001	1,78	5.025,63
characteristics	8128	2,41	51.798,37	8324	1,65	1.475,84
religious	13283	2,43	77.750,63	28947	1,71	5.096,33
artist	9836	2,53	59.242,52	70983	1,58	11.710,53
ocean	12691	2,48	73.898,57	23712	2,00	4.008,45
farming	4030	2,20	26.484,16	7720	1,85	1.352,18
achievement	6072	1,94	39.718,27	8872	6,46	1.551,72
overlooking	4947	2,32	32.406,17	4588	1,92	795,65
student	9885	2,03	59.031,49	31289	1,65	5.403,39
divisions	6505	2,07	43.971,56	11883	2,15	2.010,79
farmer	8863	2,38	53.623,72	9152	1,56	1.508,25
coined	1906	2,02	15.134,05	5756	1,63	1.062,34
reviews	2658	1,90	18.960,22	18353	1,90	3.098,52
bowling	1977	2,09	13.058,09	5999	1,82	1.017,96
compete	3148	2,14	21.191,50	17917	1,95	2.924,88
averaged	1035	2,35	7.801,03	2291	1,73	410,59
knights	5460	1,94	35.712,14	5134	1,58	897,52
muscles	7521	2,22	46.325,49	1703	1,60	332,47
electrical	2510	2,18	16.111,69	9682	1,64	1.708,61
China	9491	2,08	57.503,19	60471	1,70	9.815,01
Italy	9285	1,85	57.183,01	47264	1,57	7.936,71
Germany	8776	2,15	53.203,58	91903	1,82	15.279,72
Spain	8274	2,02	51.701,24	40307	1,88	6.712,40
eventually	7471	2,79	45.954,84	40743	1,69	7.228,11
champion	6531	2,29	43.332,04	40816	1,55	6.636,66
Mexico	7407	2,09	40.590,01	42768	1,56	7.003,96
Egypt	7221	2,54	46.606,65	12335	1,57	2.143,65
Turkey	6378	2,21	41.482,57	25181	1,70	3.927,40
grandmother	5843	1,92	36.474,36	2300	1,57	417,01
Holland	6006	2,20	39.342,01	6335	1,56	1.092,38
Russia	6200	1,91	39.130,14	37396	1,55	6.135,36

Greece	5781	1,87	37.889,38	18096	1,52	3.070,88
adjacent	5510	2,33	37.753,02	20249	1,53	3.536,24
voluntarily	5437	2,11	37.993,88	1233	1,66	234,89
massacre	4266	2,56	29.457,12	4413	1,51	800,08
devastated	1795	2,41	13.370,63	1217	1,48	240,87
psychology	2765	1,93	16.698,99	7569	1,57	1.327,04
Turkish	4294	2,08	29.729,39	18327	1,55	3.071,73
mobile	2653	2,16	17.934,20	14363	1,51	2.467,58
actress	2735	2,02	18.416,58	42937	1,63	7.152,40
Denmark	2988	2,60	21.433,54	15139	1,54	2.521,59
Norway	2861	2,31	19.659,34	25607	1,60	4.354,40
Sweden	2817	2,10	20.435,91	26934	1,58	4.463,26
anthem	1626	2,16	11.693,59	3114	1,87	572,13
broadcast	1837	2,01	12.773,96	36393	2,00	6.203,99
Jerome	1786	1,92	13.795,76	2953	1,97	541,18
Hannah	1569	2,25	11.296,40	2322	2,14	404,84
phone	1254	1,98	6.854,38	6438	1,69	1.123,41
rehabilitation	611	1,86	4.696,78	3157	1,72	548,64
spherical	1003	2,15	7.192,43	1542	1,57	288,07
Balkan	643	1,93	4.588,76	2558	1,56	454,18
Finland	803	1,92	6.533,36	15017	1,57	2.496,06
Bulgaria	703	2,32	5.506,43	10591	2,10	1.686,33
traction	548	1,95	3.579,49	1295	1,96	236,06
ethnic	375	2,40	3.139,01	13972	1,64	2.448,86
vocalist	276	2,40	1.976,54	13491	1,53	2.316,81
coastal	347	2,05	2.608,54	16430	1,90	2.787,04
television	204	2,20	1.426,55	126670	1,61	21.379,31
conceptual	118	1,96	823,79	2199	1,60	388,91
database	61	2,00	663,90	6645	1,89	1.152,88
Tchad	55	1,95	781,25	87	1,73	22,23
Edirne	18	2,18	525,86	219	1,67	70,78
Toplam Süre		**216,75**	**4.401.899,16**		**184,83**	**322.659,60**

MSSQL Server Like Arama Sonuçları

Kelime	Gutenberg			Wikipedia		
	Kelime Sayısı	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**	**Kelime Sayısı**	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**
allows	11346	500.595,21	499.845,03	19471	43.523,95	43.266,14
computer	21385	681.695,69	681.009,25	40395	42.221,72	42.235,73
student	12010	450.788,76	450.122,97	77596	42.358,41	42.390,44
Turkey	7056	637.939,73	637.684,19	26159	43.182,11	43.155,26
grandmother	6824	642.593,78	639.413,49	2570	42.461,65	42.375,16
psychology	2795	767.288,89	764.728,28	7807	42.190,42	42.231,97
Turkish	4300	715.478,43	717.171,36	18381	43.175,98	43.032,37

database	64	850.202,51	850.986,31	7915	42.507,16	42.627,57
Tchad	79	866.101,98	868.601,35	180	43.296,13	43.218,10
Edirne	22	881.664,49	881.415,54	234	43.869,89	44.054,93
Toplam Süre		**6.994.349,47**	**6.990.977,77**		**428.787,42**	**428.587,67**

MySQL Like Arama Sonuçları

Kelime	**Gutenberg**			**Wikipedia**		
	Kelime Sayısı	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**	**Kelime Sayısı**	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**
allows	11346	229.990,95	231.623,67	19471	43.788,93	43.798,76
computer	21385	234.168,85	234.145,66	40395	41.646,60	41.441,62
student	12010	222.344,83	222.672,15	77615	42.755,99	42.735,75
Turkey	7056	243.290,92	243.072,18	26160	44.419,26	44.397,99
grandmother	6824	234.533,82	231.626,73	2570	41.064,94	40.914,25
psychology	2795	242.532,37	245.523,78	7807	41.109,20	40.783,35
Turkish	4316	248.786,04	251.538,95	18383	43.994,19	44.030,39
database	64	254.452,92	255.197,22	7915	41.294,98	41.406,73
Tchad	79	263.492,93	263.272,74	180	44.226,25	44.015,58
Edirne	22	270.174,38	270.538,72	234	45.861,94	45.897,99
Toplam Süre		**2.443.768,01**	**2.449.211,80**		**430.162,28**	**429.422,41**

MongoDB Regex Arama Sonuçları

Kelime	**Gutenberg**			**Wikipedia**		
	Kelime Sayısı	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**	**Kelime Sayısı**	**Yöntem 1 Arama Süresi (ms)**	**Yöntem 2 Arama Süresi (ms)**
allows	11326	31.189,88	21.514,99	19455	11.204,14	11.244,42
computer	21385	25.690,69	17.481,00	35182	6.895,25	7.018,99
student	11736	25.300,06	18.003,44	73292	9.067,05	9.438,47
Turkey	4040	22.065,86	11.104,65	25570	4.812,16	4.998,43
grandmother	6574	14.728,82	14.387,53	2391	5.820,46	5.856,61
psychology	2542	14.875,60	14.795,25	6041	5.826,25	5.848,12
Turkish	4284	11.058,08	11.383,69	18369	4.692,99	4.804,40
database	63	21.934,75	28.038,53	7149	7.207,40	7.328,81
Tchad	56	12.590,42	11.750,45	119	4.814,64	4.891,16
Edirne	19	11.302,74	24.650,19	234	4.415,98	4.719,60
Toplam Süre		**190.736,90**	**173.109,72**		**64.756,32**	**66.149,01**

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Mahamoud BRAHIM ADOUM

EĞİTİM BİLGİLERİ

- 2020 - 2022 : Trakya Üniversitesi (Yüksek Lisans)
- 2015 - 2019 : Bilecik Şeyh Edebali Üniversitesi (Lisans)
- 2014 - 2015 : Fatih Sultan Mehmet Vakıf Üniversitesi (Türkçe Hazırlığı)
- 2010 - 2013 : Lycée de la Liberté (Lise - ÇAD)

FORMASYON

- 2022 : Cisco Certificated Network Associate (CCNA 200 - 301)

BİLEN DİLLER

- Türkçe
- Fransızca
- Arapça
- İngilizce (Başlangıç)

İŞ DENEYİMLERİ

- 2021 : Ulusal Görsel-İşitsel Medya Ofisi (Çad İletişim Bakanlığı)

TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER

Adoum M. B. & Mesut, A. (2022, Temmuz) Veri Tabanlarında Kullanılan Tam Metin Arama Yöntemlerinin Performans Karşılaştırması, In *6th Engineers of Future International Student Symposium - EFIS 2022*, Zonguldak / Türkiye.