T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

# ARAMA MOTORU MİMARİSİ VE UYGULAMASI

**Mehmet KARLIK**

**YÜKSEK LİSANS TEZİ**

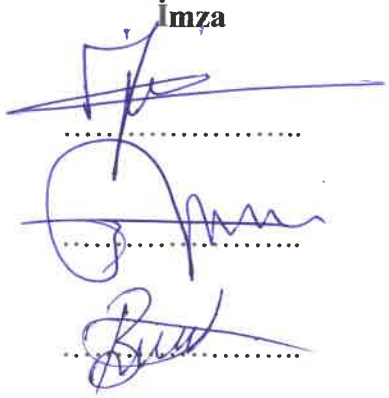
**Bilgisayar Mühendisliği Anabilim Dalı**

**Kasım-2018**
**KONYA**

## TEZ KABUL VE ONAYI

Mehmet Karlık tarafından hazırlanan "Arama Motoru Mimarisi ve Uygulaması" adlı tez çalışması 16/11/2018 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

| **Jüri Üyeleri** | **İmza** |
|---|---|
| **Başkan**<br>Doç. Dr. İsmail BABAOĞLU | ..... |
| **Danışman**<br>Dr. Öğr. Üyesi Sait Ali UYMAZ | ..... |
| **Üye**<br>Dr. Öğr. Üyesi Burak YILMAZ | ..... |



Yukarıdaki sonucu onaylarım.

Prof. Dr.
Müdür

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

**DECLARATION PAGE**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mehmet KARLIK

Tarih: 16/11/2018

# ÖZET

## YÜKSEK LİSANS TEZİ

## ARAMA MOTORU MİMARİSİ VE UYGULAMASI

**Mehmet KARLIK**

**Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü**
**Bilgisayar Mühendisliği Anabilim Dalı**

**Danışman: Dr. Öğr. Üyesi Sait Ali UYMAZ**

**2018, 137 Sayfa**

**Jüri**
**Dr. Öğr. Üyesi Sait Ali UYMAZ**
**Doç. Dr. İsmail BABAOĞLU**
**Dr. Öğr. Üyesi Burak YILMAZ**

Arama motorları internet üzerindeki büyük boyuttaki veri ile insanlar arasında köprü kuran ve insanların istedikleri bilgiye ulaşmasını sağlayan bir teknolojidir. Diğer bir deyişle arama motorları web üzerindeki internet sitelerini bot aracılığı ile kaydedip, ardından bu sayfaları indeksleyip, insanlar tarafından gönderilen sorgulara göre anlamlı veri elde edip, insanlara istedikleri bilgiyi içeren web sayfalarını gösteren bir sistemdir. Kullanıcılar çoğunlukla arama motorlarının çalışma mekanizmalarını bilmezler. Bu tez'de ilk olarak arama motorlarının kullandıkları bot yazılımı olan Crawler programlarının çalışma mimarisi anlatılarak ve paralel çalışan bir Web Crawler uygulaması yapılarak detaylı bir anlatım yapılmıştır. Ardından arama motorlarının indeksleme mimarisi anlatılmış ve paralel şekilde çalışan bir indeksleme uygulaması geliştirilmiştir. Ardından arama motorlarının indeksleme alt yapısında nasıl arama yapıldığına değinilmiş ve web arayüzüne sahip bir arama uygulaması geliştirilmiştir.

**Anahtar Kelimeler:** Arama Algoritmaları, Arama Motoru, İndeksleme Algoritmaları, Web Crawler, Pagerank Algoritması

# ABSTRACT

## MS THESIS

## SEARCH ENGINE ARCHITECTURE AND APPLICATION

**Mehmet KARLIK**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF KONYA TECHNICAL UNIVERSITY**
**THE DEGREE OF MASTER OF SCIENCE OF PHILOSOPHY**
**IN COMPUTER ENGINEERING**

**Advisor: Assist. Prof. Sait Ali UYMAZ**

**2018, 137 Pages**

**Jury**
**Assist. Prof. Sait Ali UYMAZ**
**Assoc. Prof. İsmail BABAOĞLU**
**Assist. Prof. Burak YILMAZ**

Search engines are a technology that provides the information that people who want to create a bridge between people and data on the internet. In other words, it is a system where search engines save web sites on the web through bots, then index these pages and get meaningful data according to the queries sent by people and show the web pages containing the information they want. Users often do not know the working mechanisms of search engines. In this thesis, firstly the working architecture of the Crawler programs, which are the bot software used by the search engines, is explained and a detailed description is made by a parallel Web Crawler application. Then the indexing architecture of the search engines is described and a parallel indexing application is developed. Later, a search application was developed that describes how search engines have been searched in the indexing infrastructure and has a web interface.

**Keywords:** Search Algorithms, Search Engine, Indexing Algorithms, Web Crawler, Pagerank Algorithm

# ÖNSÖZ

Projenin Araştırma-Tasarım-Geliştirme aşamalarında karşılaştığım zorluklarda benden yardımlarını esirgemeyen proje danışmanım Dr. Öğr. Üyesi Sait Ali Uymaz'a teşekkürlerimi bir borç bilirim. Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana hayatlarıyla örnek olan tüm hocalarıma saygı ve sevgilerimi sunarım.

Mehmet KARLIK
KONYA-2018

# İÇİNDEKİLER

# SİMGELER ve KISALTMALAR

## Simgeler

| | |
|---|---|
| m | Dokümanın yatay uzunluğun |
| n | Dokümanın dikey uzunluğu |
| p | Dokümanda aranılan kelimenin pozisyonu |
| Q | Sorgu Cümlesi |
| k | Terim sayısı |

## Kısaltmalar

| | |
|---|---|
| KB | Kilobayt |
| MB | Megabayt |
| s | Saniye |
| ms | Milisaniye |

## ŞEKİLLER ve ÇİZELGELER

### ŞEKİLLER

# 1. GİRİŞ

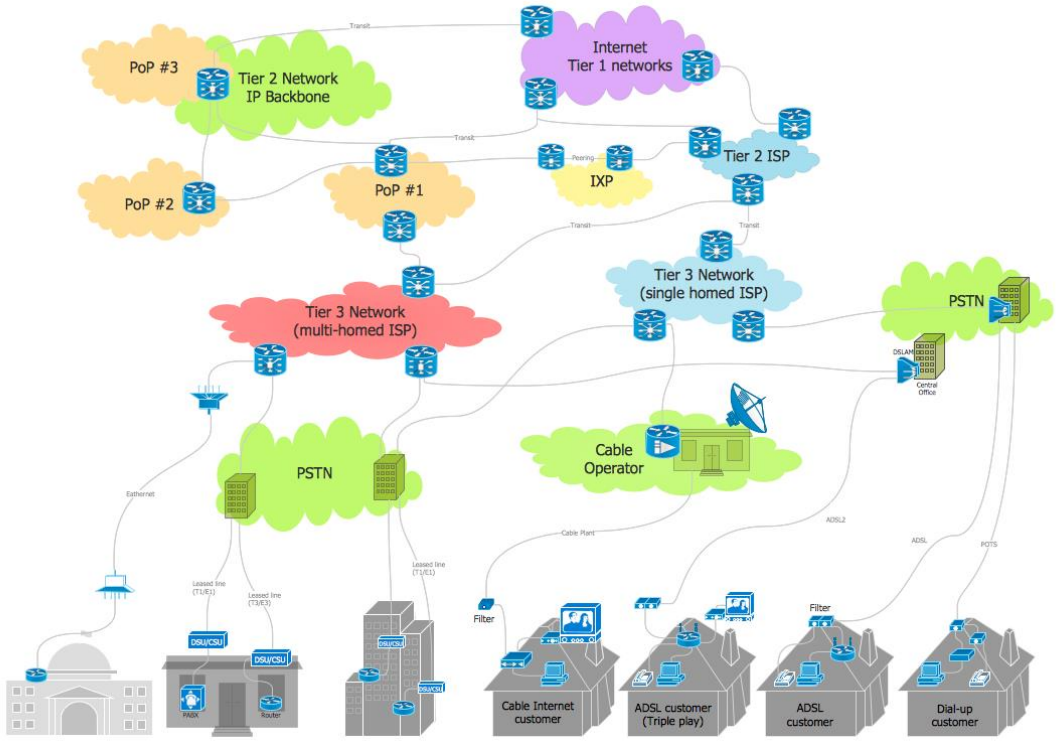
Teknolojinin ilerlediği, bilgiye en hızlı ve en kolay yoldan ulaşmanın önemli olduğu bu çağda arama motorlarının önemi oldukça büyüktür. Zamanın çok kıymetli olduğu bilindiğinden, aranılan bilginin hızlı bir şekilde bulunması büyük bir önem teşkil etmektedir. Bunu sağlayan en önemli uygulamalardan biri de arama motorlarıdır (Jain, 2013). Arama motorları, web ortamındaki aranılan bilgiyi sunabilmek adına milyarlarca web sitesinden elde ettiği verileri inceleyen ve bunları analiz ederek istenilen bilgiyi hızlı şekilde sunan önemli uygulamalardır.

Milyarlarca web sitesi verisinin içerisinde arama yapabilmek için arama motorları, kelime tabanlı arama, deyim tabanlı arama, mantıksal arama gibi birçok arama tekniği kullanmaktadır. Fakat aranılan veriler arama motorlarının kullandığı alt yapıda kayıtlı değilse, hiçbir arama tekniği kullanıcının aradığı bilgiyi sunamamaktadır. Bunun önüne geçebilmek adına, arama motorları internet üzerinde bulunan bütün web sitelerini veritabanlarına kayıt etmeye çalışmaktadırlar (Roshdi & Roohparvar, 2015).

## 1.1 İnternet

İnternet bilgisayar sistemlerini birbirine bağlayan elektronik iletişim ağıdır. İnternetin yapısı Şekil 1.1'de gösterilmektedir. Birbirine bağlı olan bilgisayarlar, donanımsal açıdan birbirlerinden farklı olsalar dahi kendi aralarında bilgi alışverişi yapabilmektedirler. Daha detaylı olarak interneti tanımlayacak olursak; TCP/IP protokolünü kullanan bütün bilgisayarlar birbirleriyle haberleşebilirler.

Ortak kullanılan ve standart hale gelen TCP/IP protokolü sayesinde dünya üzerindeki birçok kullanıcı birbirleriyle bilgi alışverişinde bulunabilmektedir. İnternet kullanıcıları, farklı bilgisayarlar kullanmalarına rağmen bu bilgisayarların birbirleriyle iletişim kurması için oluşturulan standart protokoller yardımıyla, kendi bilgisayarlarındaki bilginin erişimine izin vererek kendi aralarında bilgi alışverişi yapabilirler (Greene, 2000).

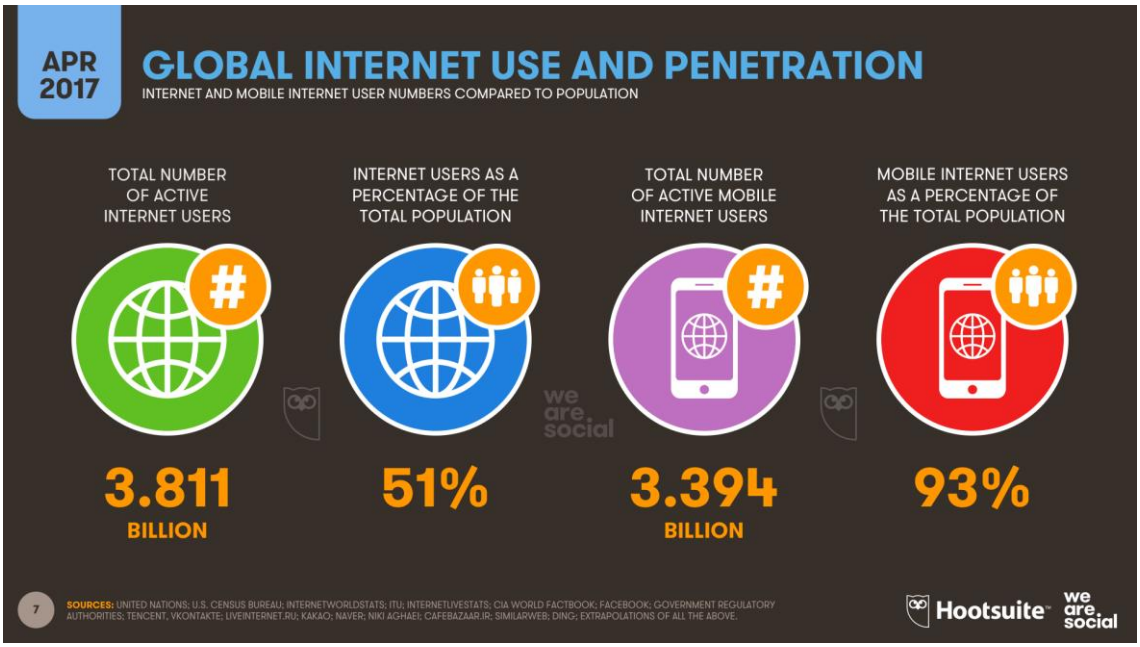



**Şekil 1.1** İnternetin Yapısı (Anonim, 2018b)

İnternete bağlı olan bilgisayar ağları ve dolayısıyla bu ağa bağlı olan bilgisayarlar, aynı zamanda internet ortamına da bağlıdır. Bu bilgisayarların bir kısmı, internet ortamında başka bilgisayarlara çeşitli bilgiler sunarlarken; diğer bilgisayarlarda internete bağlanmak isteyen kullanıcıların erişim isteklerine cevap verirler. Ancak internetin en temel kullanım nedeni, haberleşmeyi ve iletişimi sağlamasıdır. Bu sayede bilgisayarlar dünyanın dört bir tarafından çok hızlı bir şekilde birbirleriyle iletişim kurup, haberleşebilmektedirler.

## 1.2 İnternet'e Ulaşmak: (Tarayıcılar)

Tarayıcılar internette yayınlanan web sayfalarının kodlarını derleyip sizin ekranınıza yazı, resim, film ve benzeri görüntülerin gelmesini sağlayan programlardır. Tarayıcı, World Wide Web üzerinde dökümanların transfer edilip görüntülenmesini sağlayan programlara verilen addır (Gunes, 2014). Kullanıcılar web sayfalarını görüntüleyebilmek için tarayıcılar kullanmaktadırlar. Günümüzde kullanılan bilindik tarayıcılar sırasıyla; Google Chrome, Microsoft Internet Explorer, Firefox ve Safari'dir. Dünya üzerindeki aktif internet kullanıcılarının sayısı Şekil 1.2'de gösterilmiştir.




**Şekil 1.2** Dünya üzerindeki aktif internet kullanıcılarının sayısı (Kemp, 2017)

### 1.3 Sabit Kaynak Belirleyici (Uniform Resource Locator - URL)

Web üzerinde bulunan videoların, resimlerin, yazılı dokümanların, kısacası web üzerindeki her dokümanın "Uniform Resource Locator" olarak adlandırılan ve kısaltma olarak "URL" denilen bir adresi vardır. URL adreslerinin, sunucu ismi, alan adı ve protokol olarak adlandırılan parçaları bulunmaktadır. Örnek olarak, http://www.websiteismi.com sitesinin ismi değerlendirilirse: "http" bölümü URL adresinin protokol kısmını belirtmektedir ve bu protokol "Hypertext Transfer Protokol" olarak adlandırılmaktadır. Hypertext Transfer Protocol (http), internet iletişim kurallarının kısa adıdır. Bir web tarayıcısından bir sunucuya gönderilen istek http protokollerine göre gerçekleştirilir. Sunucu http protokolüne göre web tarayıcısına cevap verir. Http protokolü, tarayıcıların sunucularla haberleşmesini ve veri transfer etmesini sağlayan protokoldür. Çoğu web tarayıcısı bu protokolleri otomatik olarak adresin önüne eklerler. "https://" protokolündeki 's' secure, güvenli manasındadır ve http protokolünün güvenli halidir. Web sayfaları kullanıcılardan, kullanıcıların kişisel bilgilerini alacakları zamanlar bu protokolü kullanmaktadırlar. "www" ise (Dünyayı Saran Ağ) World Wide Web'in, kısaltılmış halidir. Alan adının uzantısı olan .com , .org gibi uzantılarda web sayfasının tipini belirtmek amacıyla oluşturulmuş uzantılardır.

Web sayfaları benzersiz bir URL adresine ve sayfaların tutulacağı bir alana sahiptirler. İnternet kullanıcıları URL adreslerini web tarayıcılarının adres satırlarına

yazarlar. Tarayıcıda bu adreste yazılı olan adresi bulmak için Domain Name Server adı verilen DNS sunucularına bu adresi gönderirler ve adresin hangi IP adresindeki bilgisayara ait olduğunu öğrenirler, sonrasında IP adresindeki host ve sunucu olarak adlandırdığımız bu bilgisayarlardan URL adresinde bulunan web sayfasının içeriklerini isterler, sunucudan alınan web sayfası tarayıcı ekranında kullanıcıya gösterilir (Segal, 1995).

### 1.4 Tipik Alan Adı Uzantıları:

- .gov – (Government) Hükümetler tarafından kullanılan alan adı uzantısı
- .edu – (Education) Üniversite gibi eğitim kurumları tarafından kullanılan alan adı uzantısı
- .org – (Organization) Organizasyon web sitelerinde kullanılan alan adı uzantısı
- .mil – (Military) Askeri web sitelerinde kullanılan alan adı uzantısı
- .com – (Company) Şirketlerde kullanılan alan adı uzantısı

### 1.5 İnternetin Tarihçesi

1962 yılında J.C.R. Licklider Massachusetts Teknoloji Enstitüsü'nde internet düşüncesini konu alan ve internetin kökünü oluşturan “Galaktik Ağ” kavramını ortaya koymuştur. Licklider, bu kavram içerisinde dünya üzerinde birbirine bağlanmış bir sistemde isteyen herkesin birbirleriyle veri ve program alışverişi sağlayabilmesini ifade etmiştir. Sonraki yıllarda MIT'de çalışan Lawrance Roberts ile Thomas Merrill, 1965 yılında bilgisayarların birbirleriyle haberleşmesini sağlamışlardır. 1966 yılı sonunda ise Roberts DARPA'da çalışmaya başlamış ve “ARPANET” isimli projesini kurula sunmuştur.

Kurul tarafından kabul edilen ARPANET sistemi ilk olarak dört merkez üzerinde çalıştırılmıştır. Bu proje ile internetin ilk şekli ortaya çıkmıştır. Kısa süre içerisinde Amerika’nın birçok merkezindeki bilgisayar sistemleri ARPANET ağına bağlanmıştır. Ardından, 1971 yılında ortak kullanılan protokollerin ilk adımı olan “NCP-Network Control Protokol” türkçesiyle Ağ Kontrol Protokolü geliştirilmiş ve sistemler bu protokol üzerinden birbirleriyle haberleşmeye başlamışlardır (McQuillan,

Richer, & Rosen, 1980). Yine bu yıl içerisinde elektronik posta (e-mail) sistemi oluşturulmuş ve bu sistem ARPANET içinde kullanılmaya başlanmıştır.

1 Ocak 1983 yılında, NCP'den daha kapsamlı özelliklere sahip yeni bir protokol geliştirilmiştir. Transmission Control Protokol, türkçesiyle Aktarım Kontrol Protokolü olarak adlandırılan ve TCP/IP olarak kısaltılan bu protokol yeni özellikleriyle birlikte ARPANET içinde kullanılmaya başlanmıştır. TCP/IP protokolü bugünkü var olan internet'in temel yapı taşı olarak kullanılmaktadır. 1990'lı yıllarda ARPANET kavramı NFSNET olarak adlandırılmış ardından sistem devlet tarafından özelleştirilerek ve genişleyerek internet bugünkü halini almıştır (Catlett, 1989) .

Günümüzde internet kullanımı her geçen gün büyük bir hızla dünyanın her köşesinde din, ülke ve dil ayrımı yapmadan artmaktadır, şu an Türkiye'de 54.3 milyon, dünyada ise 4 Milyar kadar internet kullanıcısının var olduğu bilinmektedir (Kemp, 2018)

### 1.6 Web ve İnternet Arasındaki Farklar

World Wide Web ve Internet birbirinden farklıdır. Web, dünyanın dört bir yanındaki birbirine bağlı web sayfalarından oluşmaktadır. Her sayfa metin, resim, ses klipleri, video klipler, animasyonlar ve diğer öğelerin birleşimi olabilir. Web sayfaları kendi içerisinde link olarak adlandırılan bağlantılar içermektedirler. Her bağlantı başka bir web sayfasına işaret eder ve bir bağlantıya tıklattığınızda, tarayıcınız bağlantının, gösterdiği sayfayı açar. Web'in diğer önemli özelliği, milyarlarca sayfa üzerinde arama yapabilme imkanı vermesidir. Örneğin, yaklaşık on saniye içinde, evcil kanatlı hayvan deyimini, kendi adınızı veya öğrenmek istediğiniz bir kitabın adını içeren web sayfalarının listesini alabilirsiniz. İstediğiniz bilgileri bulmak için listedeki her sayfayı görmek için bağlantıları takip edebilirsiniz.

Tarayıcının görüntülediği her sayfa, kullanıcıyı başka web sayfalarına götüren daha fazla bağlantılara sahip olabilir. Sayfalar, dünyanın her yerindeki diğer sayfalara bağlanabilir. Böylece İstanbul'dan Hong Kong'a veya New York'tan Moskova'ya kadar dünyada bulunan bütün web sayfalarını gezebilirsiniz. Genellikle, dünyanın herhangi bir yerindeki herhangi bir siteden yalnızca birkaç saniye uzaktayız. Web sayfaları arasındaki bağlantılar, sizi doğrudan ilgili bilgilere yönlendiren bağlantılar olabilir. Sayfalar arasındaki bu görünmez bağlantılar, bir örümcek ağı gibidir.

İnternet ise dünyanın her yerinden, farklı bilgisayarların ve farklı veritabanlarının kablolu veya kablosuz bağlantılar ile birbirine bağlamasından oluşan büyük bir sistemdir. World Wide Web ise internetin üzerinde çalışan bilgisayarların içerisinde kayıtlı olan belgelerin bütünüdür (Huberman & Adamic, 1999).

### 1.7 Arama Motoru Nedir?

Arama motorları web içerisinde bulunan web sayfaları ile internet kullanıcıları arasında bir köprü vazifesi görmektedir. Arama motoru, internet kullanıcılarının ihtiyaç duyduğu herhangi bir bilgiyi içeren en iyi web sayfalarını kullanıcılara listeleyen, kullanıcının aradığı bilgileri kısa sürede bulmasını sağlayan bir sistemdir. Arama motoru sırasıyla; kullanıcıdan aramak istediği kelimeyi veya metni alıp, o kelimeye veya metne uygun en yüksek alaka düzeyine sahip web sayfalarını kullanıcıya listeler. Arama motoru, "Bilgi Getirimi ve Çıkarımı" (Information Retrieval) teknolojisinin bir örneğidir.

Arama motorları web sayfalarını kaydetmek için Web Crawler olarak adlandırılan, web taraması yapan yazılımlar kullanırlar. Bu yazılımlar web sayfalarındaki URL'leri ziyaret etmek ve onları dizinlemek için çalışmaktadır. Bu sayede kullanıcı arama yaptığı zaman, bu aramadaki anahtar kelime ile bu dizine bakılarak kullanıcıya istediği anahtar kelime ile alakalı sonuçlar döndürülür.

### 1.8 Arama Motoru Tarihçesi

İlk arama motoru 1990 yılında üniversite öğrencisi olan Alan Emtage tarafından Archie adıyla kurulmuştur. Bu arama motorunun temel amacı insanların aradıkları dosyaları bulmalarına yardımcı olmaktı. 1991 yılında, Minnesota Üniversitesi'nde çalışan Mark P. McCahill, Veronica (Very Easy Rodent-Oriented Net-Wide Index to Computerized Archives ) arama motorunu geliştirmiştir. İki arama motoru da dosya aktarım mantığı üzerinde çalışmaktaydı. 1993 yılında Massachusetts Teknoloji Enstitüsü'nde çalışan ve dosya aktarım mantığını bir eksiklik olarak gören Matthew Gray, internetteki bilgileri içinde barındıran bir indeks oluşturmaya karar verip "Wandex" adındaki ilk internet botunu üreterek bu indeks sistemini oluşturmuştur. 1994 yılında da ilk üç aşamalı arama motoru olan Jumpstation geliştirilmiştir. Jumpstation'ın en büyük eksiği herhangi bir sıralama algoritması kullanmamasıydı. Aynı yıl tam metin

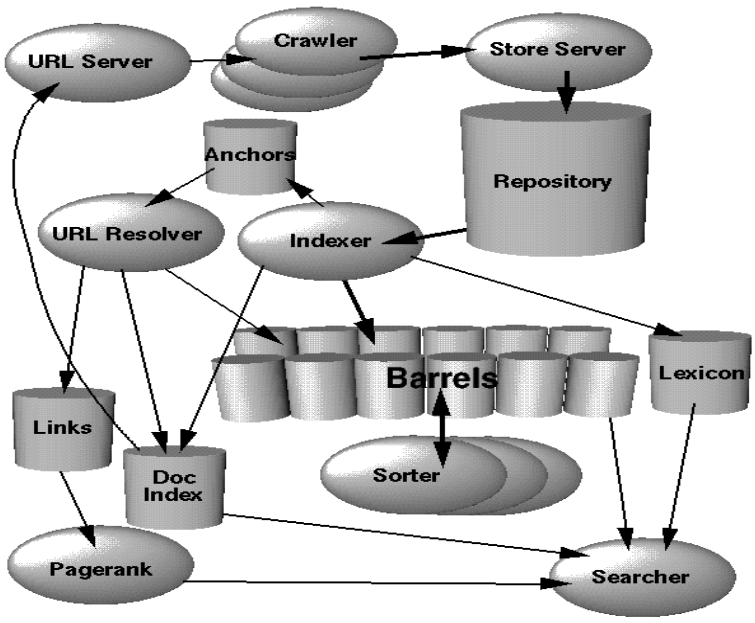
araması yapabilen WebCrawler arama motoru geliştirilirken, yine bu dönemlerde Carnegie Mellon Üniversitesi'nden Dr. Michael Mauldini tarafından geliştirilen Lycos arama motoru satışa çıkartılmıştır. Bu yıllarda kısa süre içerisinde Inktomi, Northern Light, Magellan, Excite, Infoseek ve AltaVista dâhil pek çok arama motoru geliştirilmiş ve internet kullanıcılarıyla buluşmuştur. (Anonim, 2017a)

Ancak bu arama motorları içerisinde, David Filo ve Jerry Yang'ın geliştirdiği Yahoo! arama motoru, insanların aradıkları web sayfalarını bulmaları açısından insanlar tarafından daha çok tutulup ve popüler olmuştur. Fakat 1990'ların sonlarına kadar arama motorlarına ciddi manada büyük bir yatırım yapılmamıştır.

90'ların sonlarına doğru, 1998 yılında Google'ın kurucuları olan Larry Page ve Sergey Brin, web sayfalarının anlamlı bir şekilde sıralanması için geliştirdikleri PageRank algoritmasını satmak istemişler, ancak satamamışlardır. Larry Page ve Sergey Brin bu teknolojiyi satamadıkları için büyüme kararı almış ve 1999 yılında Google arama motorunu geliştirmişlerdir. 2000 yılında popüler olmaya başlayan bu arama motoru şu anda dünyanın en popüler arama motoru olarak kullanılmaktadır (Anonim, 2017d) . Google arama motorunun mimarisi Şekil 1.3'te gösterilmektedir.

Arama motorları web sayfaları içerisindeki önemli bilgileri bulmalı ve bu bilgileri hızlı bir şekilde kullanıcılara anlamlı sonuçlar göstermek için web sayfalarını düzenleyebilmelidirler. Arama motorlarının veritabanları sürekli değiştiğinden ve geliştirildiğinden dolayı bahsedilen bu işlemler, sürekli olarak devam eden ve güncellenen bir yapıda olmaktadırlar. Dolayısıyla arama motorunun çalışmasını sağlayan yapılar iyi şekilde irdelenerek, arama motorunun veritabanı yapısı oluşturulmalıdır (Schütze, Manning, & Raghavan, 2008). Arama motorları genel itibariyle dört parçadan oluşurlar:

- Web Tarama yazılımı: Web sayfalarını gezen ve kaydeden yazılım
- İndeksleme yazılımı: Web sayfalarının içeriklerini, kelimelerine göre indeksleyip kaydeden yazılım parçası
- Puanlama yazılımı: Web sayfalarının değerini ve kalitesini belirleyen yazılım parçası
- Arama yazılımı: Kullanıcının yaptığı aramaya göre, en uygun şekilde sayfaları sıralamaya çalışan yazılım parçası




**Şekil 1.3** Üst Seviye Bir Arama Motoru Mimarisi (Brin & Page, 1998) .

Arama motorunda bir terim aratıldığı zaman, arama motoru web üzerinde çevrim içi arama yapmaz. Arama motoru, daha öncesinde veritabanına kaydetmiş olduğu veriler içerisinden arama yaparak sonuçları döndürür. Arama motoru web üzerindeki bütün web sitelerinin bir kopyasını almaya çalışır, kopyasını aldığı web sitelerinin içerisindeki bilgileri analiz eder ve kullanıcılara anlamlı sonuçlar üretmek için bu bilgileri veritabanına kaydederler. Kullanıcı arama motorunda arama yaptığında, istediği veri daha öncesinde arama motorunun veritabanında kayıtlıysa arama motoru kendi kayıtlarından bu bilgiyi çekerek kullanıcıya istediği bilgiyle alakalı olan web sayfalarını kullanıcıya listeler. Bu nedenle arama motorlarının gösterdikleri sonuçlar güncel sonuçlar olmayabilir ve web sitesi değişmiş veya kaldırılmış olabilir.

Arama motorları imkanları dahilinde web üzerinde bulunan bütün web sayfalarının tamamını kendi veritabanlarına kaydetmeye çalışırlar ama web üzerindeki bütün web sayfalarını kaydedip, indeksleyemezler çünkü burada çok büyük bir veriden bahsedilmektedir.

## 2. KAYNAK ARAŞTIRMASI

Arama motorları geliştirilirken Web Crawling, Indexing, Searching, Pagerank, Spamrank gibi birçok teknoloji birarada kullanılmaktadır. Dünyanın en büyük arama motoru olan Google'ın çalışma mimarisi, arama motoru dünyasına büyük yenilikler getirmiştir (Barroso, Dean, & Holzle, 2003).

Arama motorlarında temel olarak bilgiden anlamlı veri çıkarma teknolojisi kullanılmaktadır. Amaç, eldeki verinin içerisinden kullanıcıların aradıkları anlamlı veriyi elde etmektir. Arama motorlarının bu teknolojiyi kullanmadan önce, anlamlı bilgilerin ortaya çıkarılacağı verileri elde etmesi gerekmektedir. Bunun için, web üzerindeki sayfaların kaydedilmesini sağlayan bir program kullanılmaktadır. Bu program içerisinde "Crawling" adlı teknoloji kullanılmaktadır ve bu teknolojiyi kullanan programlara "Web Crawler" denilmektedir (Kausar, Dhaka, & Singh, 2013).

Arama motorları, bu programı birçok nedenden dolayı paralel şekilde kullanarak web üzerindeki bütün web sayfalarını kaydetmeye çalışmaktadırlar. Bu nedenlere bağlantı sorunları ve sunucu bazlı gecikmeler örnek gösterilebilir (Cho & Garcia-Molina, 2002).

İnternette bilgi toplamak okyanusta balık tutmaya benzetilmektedir. Okyanusta nasıl ki balıkçı teknelerinin ağ atabileceği derinlik kısıtlıysa internet üzerinde bilgi toplamak da kısıtlı vaziyettedir. İnternetteki üzerindeki çoğu bilgi, dinamik olarak üretilen sitelere gömülü vaziyettedir ve arama motorları bu bilgilere ulaşamamaktadır Arama motorları tarafından Web'in ulaşılamayan bu kısmına "Deep Web" (web'in karanlık yüzü) denilmektedir (Bergman, 2001) .

Web, içerik olarak gerekli gereksiz birçok bilgi (spam sayfalar, anlamlı içeriğe sahip olmayan sayfalar) içerir. Bu bilgilerin içerisinden gerekli ve düzgün olan belgelerin ayırt edilebilmesi, kullanıcılara doğru ve kaliteli sonuçlar listelemek açısından önem teşkil eder (Bernstein & Zobel, 2005). Bunun yanı sıra, Web Crawler programları aracılığıyla toplanan web sayfası verilerinin güncelliği de önemli bir konudur. Arama motorları kaydettikleri web sayfalarının güncelliğini, "Web Crawler" programları ile sağlamaktadır (Cho & Garcia-Molina, 2002) .

Veriden bilgi çıkarımı kavramına geri dönülecek olursa, anlaşılması gereken öncelikli noktalardan biri klasik veritabanı sistemleri arasındaki farklardır. Klasik veritabanı sistemlerinde, mantıksal operatörler kullanarak veriden veri elde edilirken, bilgi çıkarımı teknolojisinde ise veriden anlamlı bilgi elde edilmektedir (Van

Rijsbergen, 1979). Bu teknoloji temel olarak üç adet yapıdan oluşmaktadır. Bu yapılar sırasıyla veri analizi, veri indeksleme ve doküman sıralama işlemleridir (Schütze et al., 2008) .

Veri analizi, doküman içerisindeki kelimelerin bir sözlük haline getirilmesi işlemidir. Veri analizi işlemi ile indeksleme işlemi yapılmadan önce indeksleme yapılacak kelimelerin bir sözlüğü oluşturulmaktadır. Bu sayede indeksleme işleminde, belge içerisindeki yer alan kelimelerin ağırlıkları hesaplanılabilir ve hangi belgede hangi kelimenin kaçar adet bulunduğuna dair bir ters indeks yapısı oluşturulmaktadır. (Zobel & Moffat, 2006) .

Gerekli indeksleme işlemleri yapıldıktan sonra, bu indekslenen web sayfaları içerisinden kullanıcının aradığı bilgiyi içeren en anlamlı ve en doğru web sayfalarını listelemek için Pagerank gibi sıralama algoritmalarının kullanılması gerekmektedir. Arama motorlarının kullandığı bu algoritmalardan ve yeniliklerden birçoğunu şu anki arama motoru lideri olan Google ortaya koymuştur (Brin & Page, 1998). Google'ın getirdiği bu yenilikler içerisinde bulunan, web sitelerinin puanlanmasını sağlayan Pagerank algoritması, şu anda bütün büyük arama motorları tarafından kullanılmaktadır. Pagerank'in mantığını biraz açıklamak gerekirse, bir A sitesi B sitesinin linkini yayınlamışsa bunun nedeni, B sayfasının A sayfası ziyaretçileri tarafından dolaşılabilecek öneme sahip olduğunun düşünülmüş olmasıdır. Bu yapıya göre A sayfası B sayfasına bağlantı verdiğinden dolayı B sayfasının pagerank değerini yükseltmiş olacaktır. A sayfası ne kadar yüksek pagerank değerine sahip ise B sayfasının pagerank değeri de buna orantılı olarak artacaktır. Ayrıca A sayfasında dışarıya ne kadar az link bağlantısı verilmişse, B sayfasının pagerank değeri o kadar yüksek olacaktır (Anonim, 2012).

Yazılım geliştiriciler pagerank algoritmasının mantığını kötü niyetli olarak kullanarak kendi geliştirdikleri web sitelerini arama motorlarında üst sıraya çıkarabilmek için, popüler web sayfalarının içerisine, geliştirdikleri web sitelerinin bağlantı adreslerini yerleştirmeye çalışmaktadırlar. Bu yol ile geliştirdikleri web sitelerinin pagerank değerlerini arttırmaya ve arama motorları sonuçlarında üst sıralara çıkmaya çalışmaktadırlar. Bu tarz sayfaların belirlenebilmesi için SpamRank tarzı algoritmalar kullanılmaktadır (Benczur, Csalogany, Sarlos, & Uher, 2005). SpamRank algoritmasının içerisinde kullanılan en basit yapılardan biri, web sayfaları içerisinde bulunan yorum satırlarındaki link bağlantılarının pagerank hesaplanmasına dahil

edilmemesidir. Arama motorlarında buna benzer algoritmalar kullanılarak spam sayfaların kayıt edilmesinin ve derecelendirilmesinin önüne geçilmektedir.

Arama motorlarında kullanılan bir diğer teknolojik gelişme Caching sistemidir. Arama motorlarının hızlarının arttırılması açısından hayati öneme sahip bir teknolojidir. Kullanıcının arama motoruna girmiş olduğu bir sorgunun sonuçlarının hesaplanması ve sıralanması işlemi ciddi bir zaman kaybına neden olmaktadır. Arama motoru geliştiricileri bunun önüne geçebilmek adına önbellek sistemi oluşturmuşlardır. Önbellek sistemindeki asıl amaç, önceden sorgulanan kelime veya kelime gruplarının veritabanında sonuçlarıyla birlikte kaydetmektir. Bu sayede arama motoru, aynı kelime veya kelime grupları ile yapılan sorgularda, tekrar sorgulama ve sıralama işlemleri yapmadan, sonuçları önbellek tablosundan çekerek hızlı bir şekilde kullanıcılara sunarlar (Baeza-Yates et al., 2008).

Günümüzde, Amerika'daki bir kullanıcının arama motoru içerisinde yapmış olduğu "Restaurant" sorgusuyla, Türkiye'deki bir kullanıcının yapmış olduğu "Restaurant" sorgusunun sonuçlarının aynı olması beklenilmez. Arama motoru geliştiricileri bu sorunu çözebilmek için "Bölgesel Arama" adlı bir kavram geliştirmişlerdir. Bölgesel arama işlevini gerçekleştirmek adına da iki adet çözüm üretilmiştir. İlk çözüm, arama motorlarının her ülkeye ait farklı veritabanlarının bulunması ve ilk olarak bu veritabanlarından sorgu sonuçların üretilmesidir. İkinci olarak üretilen çözüm ise, web sayfalarının içerisindeki metinlerde kullanılan dillerin tespit edilerek bu dil ile veritabanına kaydedilmesi ve kullanıcıların bulundukları ülkeye ait ana diline göre bu dile uygun sonuçların üretilmesidir (Baker, Trinidad, & Chalkley, 2011). Ayrıca sorgu sonuçlarını arttırabilmek adına, bazı arama motorlarında sorgulama esnasında dil çeviri işlemleri de kullanılmaktadır. Arama motoru bu işlemi gerçekleştirmek için sorguyu bir dilde alıp diğer dillere çevirerek veritabanında ekstra aramalar gerçekleştirmektedir ve sorgu sonuçlarına bu aramaların sonuçlarını eklemektedir (Maeda, Sadat, Yoshikawa, & Uemura, 2000). Bu sistem, arama motorlarına işlem gücü açısından fazla maliyete neden olduğundan, genelde arama motorları açısından pek tercih edilmemektedir.

Arama motorları mimarisinde kullanılan bu teknolojik gelişmelerin ışığında, arama motorlarının ticari potansiyelinin farkedilmesinden dolayı, bazı ülkeler kendilerine ait arama motorlarını geliştirmeye karar vermişlerdir. Bu ülkelere Rusya, Çin ve Türkiye gibi ülkeler örnek verilebilir. Çin, bu adımı atan ülkelerden biridir. Bu adımı atmalarının önemli nedenlerinden biri de Google şirketinin Çin pazarına geç

girmesinden ve Çin kanunlarıyla uyum sağlayamamasından kaynaklanmıştır. Çin'deki arama motoru gereksinimine çözüm olarak Robin Li ve Eric Xu tarafından 2000 yılında Baidu adında bir arama motoru şirketi kurulmuştur (Emyan, 2014). Baidu şu anda Çin'in kullanılan en büyük arama motorudur. Ayrıca Google şirketi, Çin hükümetinin yasalarıyla uyum sağlayamadığından dolayı Çin'de kullanılması yasaklanmıştır.

Buna benzer diğer bir örneği Yandex şirketi için de verebiliriz. Yandex şirketi, Arkady Volozh tarafından 2000 yılında kurulmuştur. %56'lık internet kullanıcı sayısıyla Rusya'da en çok kullanılan arama motoru olma özelliğine sahiptir. Arkady Volozh tarafından yapılan bu atılım sayesinde Rusya'nın öz sermayesi dış ülkelere gitmemektedir (Anonim, 2017f). Buna benzer atılımları Türkiye'deki yazılım geliştiriciler atmışlarsa da henüz başarılı olamamışlardır. Şu an için Türkiye'nin yerli arama motoru olarak iki örnek bulunmaktadır. Bunlardan biri Geliyoo.com, bir diğeri de Yaani.com sitesidir.

Geliyoo.com 2010 yılında iki Türk yazılım mühendisi tarafından geliştirilmiş, gelişmiş bir arama motorudur ve hala test sürümünde kullanılmaktadır. Maddi olanakların yetersizliğinden dolayı gelişememiş ve dolayısıyla güncel veri üzerinden arama yapılamayan bir arama motorudur (Anonim, 2017c). Bir diğer arama motoru olan Yaani.com ise Turkcell tarafından geliştirilmiş yeni bir arama motorudur (Anonim, 2017e).

# 3. MATERYAL VE YÖNTEM

## 3.1 Veri Toplama

Web sayfalarında arama yapan bir arama motoru inşa etmek isteniyorsa öncelikle arama yapılacak web sayfaları elde edilmiş olmalıdır. Web sayfalarını bulmaya ve otomatik olarak indirmeye "Crawling" denilmektedir (Cho & Garcia-Molina, 2002). Crawling yapan programlara da "Web Crawler" denilmektedir.

Web Crawler programları web sayfalarını kaydederken bazı zorluklarla karşılaşırlar. Bu zorluklardan en büyüğü web üzerindeki sayfa sayısının oldukça fazla olmasıdır. Web üzerinde en az 180 katrilyon web sayfası bulunduğu düşünülmektedir (Ashley, 2018). Burada "en az" denilmesinin nedeni, hiç kimsenin web üzerinde bulunan gerçek web sayfası sayısını bilememesinden kaynaklanmaktadır. Çünkü web sayfalarının sayısı her geçen gün sürekli olarak artış göstermektedir. Dünyada her saniye bloglara ve web sitelerine yeni sayfalar eklenmektedir. Dünyadaki çoğu kurum veya arama motoru, web'in bu büyük kapasitedeki alanını kaydetmek için yeterli alana sahip değildir, sadece bazı web sağlayıcıları sürekli olarak yeni oluşturulan sayfaları kaydedip, kendi koleksiyonlarında tutmaktadırlar.

Web sayfalarının kendine özel bir URL adresleri vardır ve web sayfaları "http" protokolü kullanan web sunucularda tutulurlar. Web tarayıcıları ve Web Crawler programları iki farklı tip istemci tipinde programlardır. Her iki programda da sunuculardan web sayfaları istenirken aynı yol kullanılmaktadır. İlk olarak istemci programı, isimleri ve onlara karşılık gelen ip adreslerini kaydeden DNS sunucusuna bağlanırlar. Dns sunucusu, istemci programın gitmek istediği web sitesinin ismine bakar ve ona karşılık gelen ip adresine istemciye iletir. Ardından istemci programlar, web sitesinin kayıtlı olduğu sunucunun ip adresine bağlanmaya çalışırlar. Sunucular kendi içerisinde her biri ağı farklı bir porttan dinleyen birçok farklı program çalıştırırlar. Web sayfalarını istemci programlara sunulabilmek için ise 80 portu üzerinden istemci programlarıyla haberleşebilirler. İstemci programlar sunucuya bir kez bağlandıktan sonra sunucuya istedikleri sayfa ile ilgili bir HTTP isteği gönderirler. Buna karşılık olarak da sunucu içerisinde kayıtlı olan o sayfayı istemci programa gönderir ve istemci programın istediği sayfalar veya dokümanlar gönderildikten sonra bağlantı sonlandırılır (Castillo, 2005) .

### 3.1.2 Web Crawler

Web Crawler'lar, temelde iki göreve sahiptirler. Bunlar sırasıyla web sayfalarını indirip kaydetmek ve içindeki URL adreslerini tespit etmektir. Crawler programları genelde arka tarafta "Queue" denilen bir yapı kullanırlar. Türkçesi kuyruk olan bu veri yapısı, bir kuyruk misali çalışarak kuyruğa eklenen verilerin sırasıyla işlenmesini sağlar. Crawler programları ilk işlem olarak sayfayı indirirler, ardından sayfanın içerisindeki URL adreslerini tespit ederler. Bu URL adresleri kuyrukta mevcut değilse kuyruğa atarlar, ardından sıradaki URL adresine sahip sayfayı indirip bu iki işleme aynı şekilde sırasıyla bir döngü halinde devam ederler (Olston & Najork, 2010) .

Crawler programının kodları, tek bir Thread ile çalışacak şekilde geliştirilir ise, program etkili bir şekilde çalışmaz. Crawler programı sadece bir URL adresini indirmek için sırasıyla; DNS Server'a IP adresini sorar ve yanıt bekler, ardından web server'a bağlanmak için web server'dan yanıt gelmesini bekler, ardından sayfayı indirir. Bütün bir programı bu şekilde meşgul etmek akıllıca bir işlem değildir. Bu bekleme zamanı içerisinde CPU'nun işlem gücü ve ağ bağlantısı kullanılmamaktadır ve kullanılabilecek CPU ve ağ kaynaklarının büyük kısmı boşta beklemektedir. Programın etkili bir şekilde çalışabilmesi için çalıştığı makinanın sahip olduğu yetenekleri tam olarak kullanmalıdır. Bu yüzden, Web Crawler programları çoklu Thread çalıştıracak şekilde dizayn edilir ve saniyeler içerisinde yüzlerce sayfa indirebilen bir yapıya sahip olunur (Yadav, Sharma, Gupta, Garg, & Mahajan, 2007).

Saniyede yüzlerce sayfa indirmek arama motoru açısından faydalıdır ama web sayfalarının bulunduğu Server'lar için faydalı bir işlem değildir. Çünkü bir web Server'dan aynı anda yüzlerce sayfa indirmeye çalışmak, Web Server'ın normal kullanıcılara yanıt vermesini engelleyen bir işlemdir (Wilson, 2008). Bu problemi engellemek için Web Crawler programları bazı nezaket kurallarına sahiptirler. Bir Web Server'dan aynı zaman içerisinde birden çok web sayfası isteğinde bulunmazlar. Aynı Server'dan bir web sayfası istemeden önce en az birkaç saniye beklerler. Bu bekleme işlemi bazen dakikalar bile alabilir. Bu sayede Web Server'lar gerçek kullanıcılara yanıt vermekte zorlanmazlar ve herhangi bir sıkıntı yaşamadan Web Server'lar çalışmaya devam ederler. Bunun haricinde, bazı site sahipleri hiçbir şekilde sitelerinin Web Crawler'lar tarafından kayıt edilmesini istemeyebilirler veya bazı sayfalarının kayıt edilmesini, bazılarının da kayıt edilmemesini isteyebilirler veya kayıt yapılmasını istedikleri Google, Bing gibi arama motorlarının URL adreslerini özellikle belirterek

diğer arama motorlarının kayıt yapmasını istemeyebilirler. Bu gibi kuralları belirtmek için de robot.txt dosyasını kullanırlar. Web Crawler programlarının bir Thread'i içerisinde gerçekleştirilen işlemlere ait pseudo kod yapısı Şekil 3.1 de gösterilmektedir.

```
function CrawlThread(frontier)
        While frontier.next() do
                url = frontier.getUrl()
                if permitsCrawl(url) then
                        text = retrieveURL(url)
                        storeDocument(url,text)
                        foreach url in parse(text) do
                                frontier.addURL(url)
                        end for
                end if
        end while
end function
```

**Şekil 3.1** Web Crawler'ın Pseudo Kodu

### 3.1.3 Güncel Bilgi

Web sayfalarındaki bilgilere ekleme yapılabilir, bilgiler silinebilir veya güncellenebilir. Web sayfalarının güncel halini tutabilmek için Web Crawler programları daha önceden ziyaret etmiş oldukları sayfaları, belli zaman aralıklarında tekrar ziyaret ederler ve bu sayede kayıt altında tuttukları web sayfalarının en güncel halini kayıt altında tutmaya çalışırlar (Cho & Garcia-Molina, 2003).

Web sayfasının güncel olup olmadığını kontrol etmek için de bütün bir sayfayı karşılaştırmak yerine HTTP protokolünün kullandığı GET isteği yerine HEAD isteğini kullanırlar. HEAD isteğinde HTTP protokolü sayfayı istemciye göndermek yerine, sayfa hakkında kısa bir bilgi özeti gönderir. (Bkz. Şekil 3.2) Gelen bilgi özetinden de anlaşılacağı üzere Last-Modified özelliği kontrol edilir. Web Crawler, sayfanın güncel olup olmadığını bu özellik ile kontrol eder ve güncel değilse sayfayı tekrar indirip, sayfanın güncel halini kayıt altına alır. Bu özellik güncel ise herhangi bir işlem yapmaz, bu sayede program hem kendi ağını ve işlem gücünü, hem de Web Server'ın ağını ve işlem gücünü yormaz. Çoğu web sayfası sıklıkla değişmez, bu sayede Web Crawler programlarına ekstra bir yük binmemiş olur. Genellikle en çok değişen web sayfaları, web sitelerinin ana sayfaları veya kişisel bloglardır.


**Şekil 3.2** İstemci ve Sunucunun Haberleşmesi

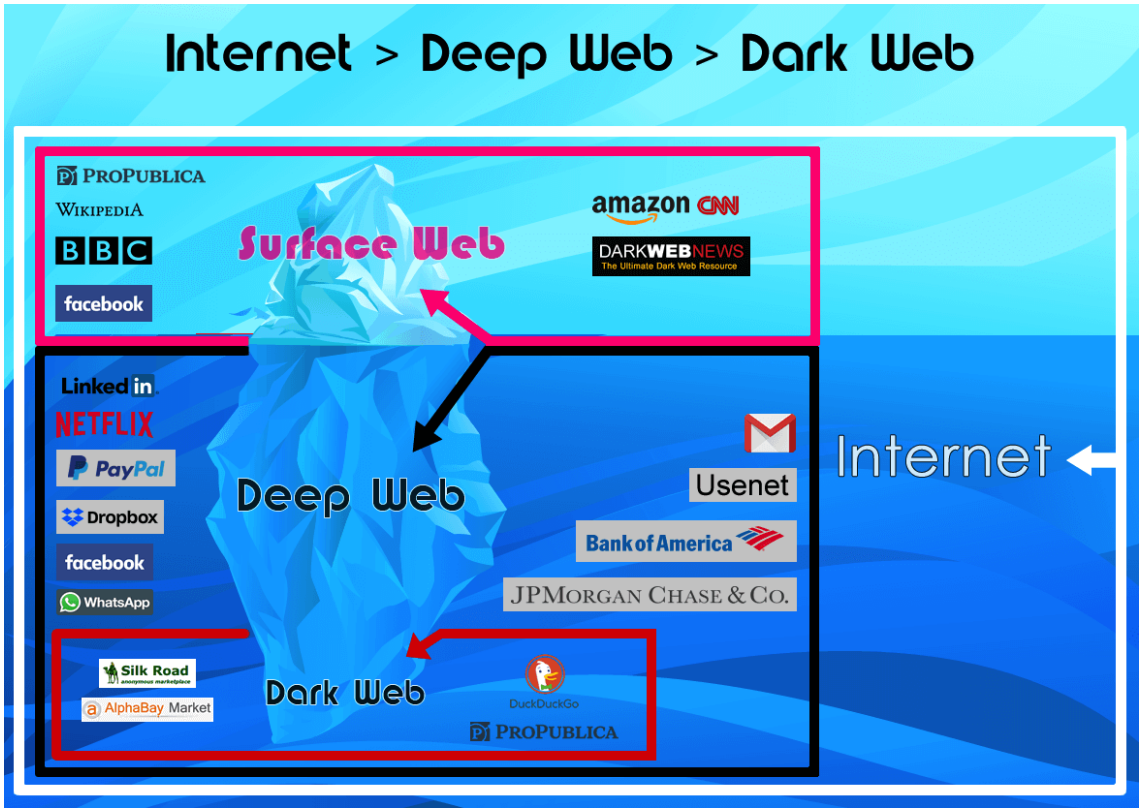
### 3.1.4 Odaklanmış Tarama (Focused Crawling)

Bazı kullanıcılar, özel bir alanda veya konuda bilgi aramak için, spesifik bir alana veya bir konuya odaklanmış özel arama motorlarını kullanmak isterler. Örnek olarak, filmler için yapılmış bir arama motorunu ele alabiliriz. Kullanıcılar filmler hakkında daha detaylı bilgi almak için filmler için özelleştirilmiş arama motorlarını kullanırlar. Bu tarz arama yapan arama motorları dikey arama denilen sistemi tercih etmektedirler. Dikey aramada, arama motoru bütün interneti kayıt altında tutmaya çalışmak yerine, sadece odaklandığı konuları içeren sayfaları kayıt altında tutar. Bu tarz arama motorları genel arama motorlarından çok daha az web sayfasını inceleyip, kayıt altına alacaklarından dolayı genel arama motorlarına kıyasla çalışma maliyetleri oldukça düşüktür (Chakrabarti, Van den Berg, & Dom, 1999). Bu tarz arama motorları bir sayfanın belirlenen konu ile ilgili olup olmadığını bulabilmek için Text Classifier yani yazı sınıflandırma teknolojisini kullanırlar.

### 3.1.5 Derin Web (Deep web)

Web Crawler'ların web içerisinde yer alan web sayfalarını taraması genellikle kolaydır. Ama bazı web sayfaları vardır ki onları taramak oldukça zor bir işlemdir. Bu tarz web sayfaları "Deep Web" yani gizli web olarak bilinen bir ağın içerisinde kayıt altında tutulurlar. Şekil 3.3'te Deep Web'in içeriğindeki yapılar gösterilmiştir.

Bazı çalışmalar, gizli web içerisinde bulunan web sayfalarının, kayıt altına alınan web sayfalarından 500 kat daha büyük bir boyuta sahip olduğunu söylemektedirler (DEEPWEBADMIN, 2016).




**Şekil 3.3** Deep Web (Anonim, 2018a)

Gizli web kendi içerisinde üç ana kategoriye ayrılır:

- Özel siteler: Bu siteler gelen linklere izin vermezler ve siteye girebilmek için özel bir kullanıcı adı ve şifreye ihtiyaç duyulur.
- Form aracılığı ile sonuçlar gösteren siteler: Bu siteler kendi içerisindeki bilgileri gösterirken form ile arama gereksinimi duyarlar. Örnek olarak otobüs ve havayolu şirketlerinin uçuş saatlerini listeleyen formları örnek gösterebiliriz.
- Script dili kullanan siteler: Bazı siteler Javascript, Flash ve diğer istemci taraflı script dilleriyle oluşturulmuştur. Web sayfasının içerisinde salt HTML kodu yerine script kodları bulunduğundan dolayı bu sayfaların içerisindeki bilgiye erişilmesi zordur (Bergman, 2001). Erişilmesi imkânsız değildir, ancak teknik açıdan zor ve Crawler uygulamasının yavaş çalışmasını sağlamaktadır. Bu yüzden bu tarz siteler indekslenmezler.

### 3.1.6 Paralel Veri Toplama (Parallel Crawling)

Web sitelerini tek tek taramak için tek bir bilgisayar yeterlidir. Ancak arama motorlarında web sitelerini kaydederken birçok bilgisayarın aynı anda veri toplaması gerekir (Cho & Garcia-Molina, 2002). Örneğin, Crawler programı saniyede 1 MB veri transfer eden bir ağ bağlantısına sahip olsun. Ortalama bir web sayfası 20 KB'lık bir veriye sahiptir. Bu da saniyede 50 sayfa kopyalanmasını sağlar. Yalnız bu şekilde hesaplama yapılırsa yanlış bir işlem yapılmış olur. Çünkü programın ilk başta Server ile bağlantı kurması gerekir. Sadece bu bağlantı kurma işlemi ortalama 80 ms'lik bir süre alır. Bağlantı kurulduktan sonra istenilen sitenin, program tarafından kayıt edilme süresi yaklaşık olarak 20 ms alır. Yani bir sitenin kayıt edilebilmesi için toplamda ortalama 80 ms + 20 ms = 100 ms'lik bir süreye ihtiyaç duyulmaktadır. Bu hesaplamada sadece ağın sağladığı veri hızı olan saniyede 1 MB'lık veri yolu ile 50 tane sitenin kaydedilme zamanı hesaplanılacak olursa; 50 çarpı 100 ms'lik bir süre yani kısacası 5 s'lik bir süre ortaya çıkmaktadır.

Fakat ağın sağlamış olduğu bütün veri indirme boyutunun tamamı kullanılmış olsaydı; ((1024 MB / 20 KB)*5) = 256 hesaplamasından görüldüğü toplamda 5 saniye içerisinde 256 adet sayfa kayıt edilebilecekti. Özetlenecek olursa, saniyede 1 MB'lık veri indirme hızına sahip bir ağda, paralel olarak çalışan bir Web Crawler uygulaması 1 saniye içerisinde 256 sayfa kayıt ederken, paralel olarak çalışmayan bir Web Crawler uygulaması 1 saniyede sadece 50 adet web sayfası kayıt edilebilmektedir (Croft, Metzler, & Strohman, 2010). Bu örnekten anlaşılacağı üzere, Web Crawler programları geliştirilirken, kendi içlerinde birden çok program parçacığı kullanılacak şekilde geliştirilirlerse, bulundukları ağları daha verimli ve etkin şekilde kullanmaları sağlanılabilmektedir.

## 3.2 İndeksleme Mimarisi

### 3.2.1 Metin İşleme

Arama motorları, arama yapılmak istenilen web sitelerini topladıktan sonra, bu toplanmış web sitelerinde arama yapabilmesi için, web sayfalarını uygun bir şekilde yeniden yapılandırması gerekmektedir. Bu yeniden yapılandırılma evresi, "metin işleme" evresi olarak adlandırılmaktadır. Metin işleme evresinin temel amacı, web

sitelerinin içerisindeki formlarda bulunan yazıları kelimelere ayırmak ve ayırdıkları bu kelimelerin doküman içerisindeki frekansını bulmaktır.

Metin işleme evresinin uygulanmasının temel nedeni, doküman içerisinde anlamlı bir arama yapmaktır (Büttcher, Clarke, & Cormack, 2016). Metin işleme ve indeksleme işlemi yapılmadan doküman içerisinde arama yapıldığında, bu işlemlerin neden gerekli olduğu anlaşılabilir. "Word", "Notepad" gibi metin düzenleme yazılımları kelime arama işlevi içerisinde, girilen kelimeyi ararken, arka tarafta metin işleme evresi kullanmadan arama yapmaktadırlar. Bu tarz programlar içerisinde arama yapılırken, aranılan kelime veya cümle, metin içerisindeki bütün kelimeler sırasıyla gezilerek, bulunmaya çalışılmaktadır. Kullanıcıların aradıkları kelimeleri bulabilmeleri için, kelimeyi veya cümleyi doğru olarak girmeleri gerekmektedir. Örneğin; kullanıcılar bir metin içerisinde "Bilgisayar Donanımı" araması yapıyorsa, program metinin içerisinde yer alan "bilgisayar, donanımı" veya "bilgisayar mimarisi ve donanımı" cümleciklerini bulamazlar. Çünkü metin düzenleme yazılımları metin içerisindeki satırları sırasıyla tarayarak, aranılan kelime veya cümle ile tamamen aynı olan sözcük veya sözcük gruplarını bulmaya çalışırlar. Arka tarafta metin işleme ve indeksleme işlemi kullanılmadığından dolayı yeterli bilgi elde edemezler (Larson, 2010).

Arama motorları metin işleme evresini kullandıklarından dolayı, metin içerisindeki kelimeleri noktalama işaretlerinden ayıklayabilirler. Kelimelerin bu şekilde kelime kelime ayrılma ve noktalama işaretlerinden temizlenme işlemine "Tokenization" işlemi denilmektedir. Bu işlem uygulanırken, sorgu işlemlerini daha etkili ve verimli hale getirmek için bazı kelimeler tamamen göz ardı edilirler. Bu göz ardı etme işlemine ise "Stopping" işlemi denilmektedir. Stopping işlemi yapılırken de göz ardı edilen kelimelere "Stopwords" denilmektedir. Bu metin işleme teknikleri kullanılarak arama motorlarının daha anlamlı sonuçlar üretmesi sağlanmaktadır. Ayrıca metin işleme teknikleri kullanılması sayesinde bilgisayarların metinleri anlayabilmesi için karmaşık mantık algoritmalarının kullanılması gerekmez. Metinlerin sözdizimsel ve anlamsal analizini içeren gelişmiş doğal dil işleme teknikleri bulunmaktadır. Ancak bu tekniklerin, bugüne kadar arama motorlarının kullandığı sıralama algoritmalarında çok az etkileri olmuştur.

Metin işleme evresinde yapılan bir diğer işlem ise kelimelere ayrılan dokümanın içerisinde her bir kelimeden kaçar adet bulunduğunun hesaplanmasıdır. Bu hesaplama işlemi vasıtasıyla kelimelerin doküman içerisindeki adedi bulunmaktadır. Kelimelerin doküman içerisindeki adedine, "frekans" değeri denilmektedir (Croft et al., 2010).

### 3.2.2 Metin İstatistikleri

Diller, inanılmaz derecede zengin ve çeşitli olmasına rağmen, aynı zamanda çok tahmin edilebilirlerdir. Belirli bir konuyu veya olayı tanımlamanın birçok yolu vardır, ancak olayların tanımlanmasında rol oynayan kelimelerin adedi sayılırsa, bazı kelimelerin diğer kelimelerden çok daha sık bir şekilde ortaya çıktığı anlaşılmaktadır. Örnek verilecek olunursa, "ve", "veya" kelimeleri bu sık kullanılan kelimelerden bazılarıdır. Bu tarz kelimeler incelendiğinde anlatılmak istenen olaylarda pek bir rol oynamadıkları anlaşılmaktadır. Stopwords kelimeleri bulunurken de, bu tarz istatistiklerden faydalanılarak kelime ayrıştırma işlemleri gerçekleştirilmektedir.

### 3.2.3 Belge Ayrıştırma

Web sayfalarında o sayfa ile alakalı anlamlı kelimelerin dışında, meta veriler , HTML ve Javascript kodları gibi birçok farklı türde içerikler olabilir. Meta veriler, metin içeriğinin bir parçası olmayan ve sayfanın oluşturulma tarihi ve yazarı gibi bilgiler içeren verilerdir. Belge Ayrıştırıcıları, web sayfalarının içerisinde bulunan meta verilerini, HTML ve Javascript kodlarını temizlemek için kullanılırlar. Bir belge, belge ayrıştırma işleminden geçtikten sonra, geriye sadece salt içerik kalır. Bu sayede, içerikte rahatça indeksleme işlemleri yapılabilmektedir.

#### 3.2.3.1 Kelime Ayrıştırma

Kelime, bir boşluk veya başka bir özel karakterle sonlandırılmış iki veya ikiden fazla uzunlukta alfa nümerik karakter dizisi olarak tanımlanmaktadır. Belge ayrıştırma işleminin ardından kelimeler, boşluk karakterlerine bakılarak ayrıştırma işlemine sokulup, bir dizi içine konulmaktadır. Kelimeler ayrıştırıldıktan sonra tüm büyük harfler küçük harfe dönüştürülür.

Sorgular için yapılan kelime ayrıştırma işleminin, belgeler için kullanılanla ile aynı olması gerekir. Sorgular ve dokümanlar için farklı kelime ayrıştırma süreçleri kullanılıyorsa, dokümanlardaki terimler ile sorgulardan gelen terimler birbirleriyle eşleşmezler. Örneğin; söz dizimi hatalarına bakılmaksızın kelime ayrıştırma işlemi yapılırsa, sorgulama yapılan kelime veya kelime gruplarında da herhangi bir hata düzenleme işlemi yapılmaması gerekir. Eğer dokümanın kelime ayrıştırma işleminde

hataları düzeltme işlemi yapılıyorsa, bu işlem sorgulanacak kelime ve kelime gruplarında da tekrar edilmelidir.

Kelime ayrıştırma yaparken noktalama işaretlerini metinden ayırmak önemlidir. Örnek vermek gerekirse BMW markasının "3.20" modeli aracı kimi dokümanda "320" kimi dokümanda "3.20" olarak yer alır. Bu iki kelime aynı anlama geldiğinden dolayı kelime ayrıştırma işleminden önce noktalama işaretlerinin kaldırılması büyük önem teşkil etmektedir (Fujii & Ishikawa, 1999) .

### 3.2.3.2 Stopping

Cümlelerde kullanılan bazı bağlaç ve kelimelerin, o cümleye kattıkları anlam düşüktür. "O", "bu", "şu", "ve", "veya" gibi kelimeler bu yapılara örnek olarak verilebilirler. Bu kelimeler metinlerde oldukça yoğun şekilde tekrarlanmaktadır. Hem ortaklık, hem de işlevleri nedeniyle, bu kelimeler nadiren kendi başlarına belge hakkında bir şey ifade ederler. Dolayısıyla arama motorları bu kelimeleri indekslemekten kaçınırlar. Bu kelimeler indekslenirse diskte gereksiz yere büyük yer kaplamış olacaktır. Bu tarz sürekli tekrar eden ve cümlede belirgin bir anlam ifade etmeyen kelimelere "Stopwords" denilmektedir. Onlara "Stopwords" denilmesinin diğer bir nedeni de kelime ayrıştırma işlemi içerisinde görüldüklerinde işlem durur ve dizi içerisinden dışarı atılırlar. Bu kelimeleri atmak, bir yandan indeks boyutunu azaltırken, diğer yandan da arama motorlarının verimliliğini arttırmaktadır.

Stopword listesi oluşturulurken dikkat edilmelidir. Metin içinden çok fazla kelimeyi kaldırmak, kullanıcıların bulmak istedikleri bilgileri içeren web sitelerine ulaşmayı engeller. Stopword listesi, kullanılacağı dildeki frekans değeri yüksek kelimeleri tespit ederek de oluşturulabilmektedir. Bu kelimelerin kullanıldıkları dil içerisinde frekans değerleri, diğer kelimelere göre oldukça yüksektir. Fakat bu işlem, bazı sorgular için önemli olan sözcüklerin dâhil edilmemesine de yol açabilir. O yüzden daha tipik bir durdurma listesinin belirlenmesi ve kullanılması gerekir. Arama motoru tarafından kullanılan depolama alanı, yeterli büyüklüğe sahipse, toplanan belgelerdeki tüm sözcükleri dizine eklemek en iyisidir. Bir dizindeki tüm Stopwords'lerin tutulması alan gereksinimlerinden dolayı mümkün değilse, maksimum esnekliği sağlamak için mümkün olduğunca az sayıda kelime çıkartılmalıdır (Blanchard, 2007).

### 3.2.3.3 Kök Bulma

Kök bulma işlemi, adından da anlaşılacağı üzere, kelimelerin köklerinin bulunması işlemidir. Ayrıştırılan kelimeler, olduğu gibi değil de, kökleriyle beraber kayıt edilirler ve kullanıcıların indeks dizininde yapmış olduğu sorgular da bu kök bulma işleminden geçirildikten sonra, indeks dizininde arama işlemi gerçekleştirilir. Genel olarak, metin ile arama uygulamaları için "kök bulma" işlemi kullanılır. Böylece sonuçların kalitesinde küçük ama fark edilebilir bir iyileşme sağlanabilmektedir. Arapça, Rusça yada Türkçe gibi sondan eklemeli dilleri içeren uygulamalarda ise arama sisteminin etkili bir parçası değildir. Sondan eklemeli dillerde, kelimelerin sonlarına yapılan eklentiler büyük anlam değişikliklerine neden olmaktadır (Zhai & Lafferty, 2004).

Algoritmik ve sözlük tabanlı olmak üzere iki temel kök bulucu tipi vardır. Algoritmik kök bulucular, belirli bir dile ait kelimelerin, soneklerinin bilgisine dayanarak kök bulma işlemi gerçekleştiren program parçalarıdır. Buna karşılık, sözlük tabanlı kök bulucuların kullandıkları mantıksal bir algoritmaları yoktur. Bunun yerine, kelimelerin köklerini bulurken, önceden oluşturulmuş sözlüklere bakarak, kök bulma işlemi gerçekleştirirler.

İngilizce dilinde kullanılan algoritmik kök bulucuların en basit türü, sonek kök bulucusudur. Bu tür bir kök bulucu, "s" harfi ile biten kelimelerin çoğul olduğunu varsayar ve böylece "books → book", "pencils → pencil" şeklinde kök bulma işlemini gerçekletirirler. Elbette, bu kural mükemmel değildir. Çünkü "century" ve "centuries" gibi birçok çoğul ilişkiyi tespit edemezler. İngilizcede kullanılan daha karmaşık algoritmik kök bulucular, -ing veya -ed gibi daha fazla türde ekleri dikkate alarak kök bulma işlemi gerçekleştirirler.

Sözlük tabanlı bir kök bulucu, hataların ortaya çıkması sorununa farklı bir yaklaşım sunar. Harf kalıplarından kelime ilişkilerini tespit etmeye çalışmak yerine, ilgili kelimelerin listeleri büyük bir sözlükte saklanır. Bu kelime listeleri insanlar tarafından oluşturulacağından algoritmik kök bulucuların yaptıkları yanlış işlemlerin önüne geçilmektedir.

Kök bulma işleminde bir diğer strateji ise, algoritmik bir kök bulucu ile sözlük tabanlı bir kök bulucuyu bir arada kullanmaktır. İlk başta sözlük kök bulucu kelime tespiti yapılır, ardından algoritmik kök bulucuyla da bu sözcükleri eklerinden ayırma işlemleri gerçekleştirilir.

Bazı uygulamalarda, hem esnekliliği hem de verimliği sağlamak için kelimeler kökleri ile indekslenirler. Daha önce belirtildiği gibi, Türkçe gibi bazı diller için kelimeleri ve kökleri ayrı ayrı indekslemek önemli olabilir. Dile özgü arama algoritmalarını gerçekleştirmek için, birden fazla dil için bir arama motorunu özelleştirmenin veya ulusallaştırmanın önemi büyüktür. Türkçe kelimeleri köklerine indirgeyen bir algoritma, yapay zeka teknikleri kullanılmayacaksa, arama motoru için kayda değer ölçüde bir yarar sağlamaz. Çünkü Türkçe geniş sonek potansiyeline sahip bir dildir. Bu yüzden arama motorlarının bölgesel olarak, bulunduğu bölgenin dil yapısına uygun çalışmaları gerekir (Chen & Gey, 2002) .

### 3.2.4 Belge Yapısı ve İşaretleme

Web arama motorları söz konusu olduğunda, sorgular belge yapısına veya alanlarına değinmez. Ancak bu yapının önemsiz olduğu anlamınada gelmez. HTML biçimlendirmesiyle gösterilen web sayfası yapılarının bazı kısımları, sıralama algoritmaları tarafından kullanılan çok önemli özelliklere sahiptir. Bu yüzden belge ayrıştırıcısı bu yapıyı tanımalı ve indeks oluşturmaya uygun hale getirmelidir.

Web sayfalarının içerdikleri metinlerin başlıkları ve HTML elementlerinin bazı öznitelikleri sıralama alıgoritması için oldukça önemlidir. Bu başlıklar HTML içerisinde <h1> , </h1> elementleri arasında yer alırlar. Elementlerde, “attribute_name” = ”value” çiftleri tarafından verilen özniteliklere (değerlerle) sahip olabilir. Bu değerlerde sıralama algoritmaları açısından önem teşkil ederler. Örneğin, bir HTML belgesinin <head> öğesi, tarayıcı tarafından görüntülenmeyen meta verileri içerir. Web sayfasına ait anahtar kelimeleri içeren meta verisi <meta name = ”anahtar kelimeler”/> şeklinde belirtilir. Bu anahtar kelimeler de sıralama yapılırken kullanılan değerlerdir. Ayrıca <head> elementi içerisinde bulunan <title> </title> elementinin içerisinde web sayfasının başlığı yer alır. Bu alan da sıralama algoritmasının kullandığı önemli değerlerden biridir.

HTML belgesinin <body> öğesi ise görüntülenen içeriği içermektedir. Ana başlık <h1> etiketi ile belirtilir. Önem taşıyan diğer başlıklar, <h2>,<h3>,<h4>,<h5>, <h6> etiketleriyle belirtilir ve bu etiketler başlığın farklı boyutlarda gösterilmesini sağlarlar. Ayrıca, <a href=”https://www.youtube.com/feed/trending” title=”Trend Listesi”> Youtube Trendler </a> gibi bağlantılar da sayfa içerisinde çok yaygındır. Bu bağlantılar, PageRank gibi bağlantı analiz algoritmalarının temelini oluştururlar.

Bağlantılar ve bağlantı metni, web araması için özellikle önemlidir. Çünkü bir bağlantıya ilişkin başlık özniteliği, bu bağlantı hakkında daha fazla bilgi sağlamak için kullanılır. Örneğin, Şekil 3.4'teki web sayfasının aşağıda belirtilen sayfa URL'si incelenirse:

***"http://www.milliyet.com.tr/kristof-kolomb-kimdir--kristof-kolomb-neyi-kesfetmistir--molatik-806/"***

"Kristof" ve "Kolomb" kelimelerinin URL'de yer alması, bu sayfadaki kelimelerin önemini arttıracaktır.

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>
            Kristof Kolomb kimdir? Kristof Kolomb neyi keşfetmiştir?
        </title>
        <meta name="Description" content="Kristof Kolomb kimdir? Kristof Kolomb neyi icat
        etmiş ya da neyi keşfetmiştir? " />
    </head>
    <body>
        <div>
            <h1 class="title">Kristof Kolomb kimdir? Kristof Kolomb neyi keşfetmiştir?</h1>
            <span>Kristof Kolomb kimdir? Kristof Kolomb neyi keşfetmiştir?</span>
            <div class="mDetailSpot">
                Kristof Kolomb kimdir? Kristof Kolomb neyi icat etmiş ya da neyi keşfetmiştir?
            </div>
            <div class="image">
                <img src="
                http://idora.milliyet.com.tr/MolaSlider/2018/01/08/fft351_mf26593682.Jpeg"
                alt="Kristof Kolomb kimdir? Kristof Kolomb neyi keşfetmiştir?" />
            </div>
            <div class="postItem">
            <div class="summary">
            <p>
                31 Ekim 1451 tarihinde dünyaya gelen Kristof Kolomb İtalyan kaşif, denizci
                ve sömürgecidir. Ceneviz Cumhuriyeti'nde doğan Kristof Kolomb, İspanya
                Katolik Kralları için hizmet etmiştir. Atlas Okyanusu boyunca 4 defa gemi
                ile seyahat etmiştir.
            </p>
            <p>
                <strong>
                    Kristof Kolomb neyi keşfetmiştir?
                </strong>
            </p>
            <p>
                Avrupa kıtasındaki krallıklar ekonomi ve emperyalizm yarışı içerisinde yeni
                ticaret yolları ve sömürge arayışındayken Kristof Kolomb'tan Doğu Hint
                Adaları'na seyahat etmesi istendi. Bu seyahat baharat yoluna bir şekilde
                girmeyi amaçlayan İspanya Krallığı tarafından da desteklendi.
            </p>
        </div>
    </body>
</html>
```

**Şekil 3.4** "http://www.milliyet.com.tr/kristof-kolomb-kimdir--kristof-kolomb-neyi-kesfetmistir--molatik-806/" sayfasının HTML kaynak kodları

### 3.2.5 Bağlantı Analizi

Sayfaları bağlayan bağlantılar (linkler), Web'in önemli bir bileşenidir. Bağlantılar, Web'de gezinen kişiler için güçlü bir gezinme yardımcısıdır. Bunun yanı sıra, arama motorlarının sayfalar arasındaki ilişkileri anlamalarına da yardımcı olurlar. Algılanan bu ilişkiler, arama motorlarının web sayfalarını daha etkili bir şekilde sıralamasına yardımcı olurlar. Bu bağlantı linkleri web tarayıcınızda göründüğünde, genellikle altı çizili ve farklı bir renkte, normal metinden farklı olarak görüntülenirler.

#### 3.2.5.1 PageRank

Dünyada milyarlarca web sayfası vardır fakat bunların çoğu ya önemli bir içeriğe sahip değildir ya da çoğu spam sayfalardır. Örneğin; kişisel bloglar, küçük bir kitleye sahipken, buna karşılık haber siteleri ve popüler şirketlerin web siteleri, birçok yararlı sayfaya sahip olabilmektedirler. Bundan dolayı sayfaların popülaritesi, arama motorlarının sıralama yapabilmesi açısından büyük bir önem teşkil etmektedir.

Web'in büyüklüğü, arama motorları için büyük bir sorun teşkil etmektedir. Örneğin; "www.sahibinden.com" sitesinin ziyaret edilmek istendiği varsayılırsa ve "www.sahibinden.com"'un URL adresi bilinmiyorsa, bir arama motoruna "sahibinden" kelimesi yazılabilir, ancak "sahibinden" kelimesini içeren milyonlarca web sayfası vardır. Kaliteli bir arama motorunun bu sonuçlar içerisinden en popüler (ve muhtemelen en doğru) olanları seçip sıralaması gerekmektedir. Aynı zamanda, bu doğru sıralamayı yaparken de bazı sıralama algoritmaları kullanılmaktadır.

Günümüzde, sıralama algoritmaları arasında en çok konuşulan algoritma, Google şirketinin kendi arama motoruyla ilişkilendirdiği "PageRank" algoritmasıdır. Genel olarak web sayfaları içerisindeki bağlantılar, popüler sayfaları işaret etme eğilimindedirler. Pagerank algoritmasıda bu mantık üzerine kurulmuş bir algoritmadır.

İnternet'teki her web sayfasının bir "Pagerank Değeri" vardır ve bu değer web sayfalarındaki bağlantılar ile belirlenir. Pagerank algoritması, popüler sayfalar ve popüler olmayan sayfalar arasında ayrım yapma yeteneğine sahip bir algoritmadır. Web sayfasının pagerank değeri, arama motorlarının en popüler web sayfasını bulabilmesi için, örneğin "sahibinden" sözcüğünü içeren milyonlarca sayfanın sıralanmasında, arama motoruna yardımcı olmaktadır. Özetlenirse, Pagerank algoritması, web arama motorları için sıralama kabiliyetini arttırabilen önemli bir algoritma örneğidir (Brin &

Page, 1998). Pagerank algoritmasını kullanan arama motorları, web sayfalarını sıralarken yüksek pagerank değerine göre sıralama işlemi yaparlar. Bununla birlikte, pagerank algoritması, arama motorlarının sıralama yaparken kullandıkları birçok özellikten sadece bir tanesidir.

Pagerank'a göre, bir A sitesi B sitesinin linkini yayınlamışsa bunun nedeni B sayfasının A sayfası ziyaretçileri tarafından dolaşılabilecek öneme sahip olduğu düşünülmüş olmasıdır. Bu yapıya göre A sayfası B sayfasına bağlantı verdiğinden, B sayfasının pagerank değerini yükseltmiş olacaktır. A sayfası ne kadar yüksek pagerank değerine sahip ise B sayfasının pagerank değeri de buna orantılı olarak artacaktır. Ayrıca A sayfasında dışarıya ne kadar az link bağlantısı verilmişse, B sayfasının pagerank değeri o kadar yüksek olacaktır (Anonim, 2012). Pagerank algoritmasının genel formülü denklem 3.1'de gösterilmektedir:

$$PR(A) = (1-d) + d( PR(t1) / C(t1) + PR(t2) / C(t2) + ... + PR(tn) / C(tn)) \quad (3.1)$$

Denklemdeki terimlerin tanımları aşağıda verilmiştir.

- **t1…tn:** Pagerank değerini bulmak istediğimiz sayfaya bağlantı veren sayfalar
- **PR(tn):** Bağlantı veren sayfaların kendi değerleri
- **C(tn):** Her sayfanın diğer sayfalara verdiği bağlantı sayısı
- **d:** 0-1 arasında kaydedilen sönüm katsayısını ifade eder. Sönüm katsayısı d genelde 0.85 olarak alınır.
- **1-d:** 1-d değeri, web sitelerinin başlangıç pagerank değerine sahip olabilmesi için oluşturulmuş bir değerdir. Örneğin, "d" katsayısı 0.85 alınan formüllerde websitelerinin başlangıç pagerank değerleri "1-d" değeri sayesinde 0.15 olarak hesaplanmaktadır.

#### 3.2.5.2 Bağlantı Kalitesi

Pagerank gibi algoritmaların ticari arama motorlarında kullanıldığı bilindiğinden, web sayfalarının tasarımcıları, geliştirdikleri web sitelerinin, arama motorlarındaki sıralamalarını, iyileştirmek için başka popüler web sitelerinde bağlantılar oluşturmaya çalışmaktadırlar. Bu işleme "Spam Link" denilmektedir. Örneğin, web tasarımcıların popüler blogların yorumlar bölümüne, tasarladıkları sayfanın linkini eklemeleri bir spam link örneğidir (Benczur et al., 2005). Popüler bir web sitesi

içerisinde bulunan bağlantının pagerank değerinin yüksek olacağı bilindiğinden, popüler websitelerinin yorum bölümleri, spam link gönderenler için çekici bir hedeftir. Bundan dolayı, arama motoru şirketleri bu yorum bölümlerindeki bağlantıları otomatik olarak algılarlar ve pagerank değeri hesaplamasında bağlantıları etkin bir şekilde göz ardı ederler. Arama motorlarına yardımcı olmak amacıyla bazı web sitesi sahipleri, arama motorlarının bu tarz spam linkleri algılayabilmesi için önemsiz bağlantılara "rel=nofollow" niteliğini otomatik şekilde eklemektedirler. Çoğu blog yazılımı, blog yorumundaki linkleri otomatik olarak "rel = nofollow" özniteliği ile işaretlemektedir. Bağlantılar hala blogda görünürler, ancak arama motorları "rel = nofollow" olarak işaretlenmiş tüm bağlantıları yok saymak için tasarlandığından, bu linkleri pagerank hesaplamasına dahil etmezler. Bu tarz işlemler, PageRank hesaplamasının bütünlüğünü korumaya yardımcı olur.

### 3.2.6 Ulusallaştırma

Web sadece İngilizce konuşanlar tarafından değil, tüm dünyada kullanılmaktadır. Web'in %65 - %70'i İngilizce yazılsa da bu oran düşmeye devam etmektedir. Bu nedenle Web'de arama yapan kişilerin yarısından fazlası İngilizceyi birincil dili olarak kullanmazlar.

Çoğunlukla Türkçe konuşan kullanıcılara sahip bir ortam için tasarlanan bir uygulama bile koleksiyonda pek çok Türkçe olmayan belgeye sahip olabilir. Örneğin, herhangi bir Web arama motorunda bir sorgu olarak "pôle nord" (kuzey kutbu) sorgusunu kullanmayı denerseniz, kaç tane Fransızca web sayfası listelendiğini görebilirsiniz. Tek bir dil kullanan arama motoru, adından da anlaşılacağı gibi, belirli bir dil için tasarlanmış bir arama motorudur. İndeksleme teknikleri ve geri çağırma modellerinin çoğu, herhangi bir dil için spesifik olarak çalışmaktadır (Tong, Lerner, Singhal, Haahr, & Baker, 2008). Örneğin, Türkçe dili için kullanılan indeksleme teknikleriyle, İngilizce dili için kullanılan indeksleme teknikleri aynı değildir. Arama motoru tasarımında en fazla etkiye sahip olan diller arasındaki farklar, arama için kullanılacak indeks dizini oluşturulurken kullanılacak metin işleme adımlarıyla alakalıdır. Bu sayede arama motorları, yapılan aramalarda ilgili sonuçların onda dokuzunun, yapılan arama sorgusuyla ilgili sonuçlar olmasını sağlamaktadırlar.

### 3.2.7 İndeksleme

Öğelerin bir listesi saklanılmak isteniyorsa, bağlantılı listelerini ve dizileri kullanmak gerekebilir. Bir bilgisayar ile yapılmak istenen çoğu şey diziler ile yapılabilmektedir. Ancak dizilerin birçok dezavantajı vardır. Örnek vermek gerekirse, sıralanmamış diziler arama algoritmalarında oldukça yavaştır ve sıralanmış diziler de veri ekleme işlemleri için oldukça yavaştır. Buna karşılık, karma tablolar ve ağaç yapıları hem arama hem de ekleme algoritmaları için belirgin şekilde hızlı bir yapıya sahiptirler. Fakat bu yapılar dizilere göre daha karmaşıktırlar. Bu yapıların yetersizliğinden dolayı metin arama algoritmaları normal algoritmalardan oldukça farklı bir işlemle hareket ederler. Metin arama algoritmaları genelde "Ters İndeks" adında bir yapı kullanmaktadırlar (Bacak, 2016) . Metin arama algoritmalarında kullanılan bu ters indeks yapısı tüm modern web arama motorlarının merkezindeki yapı olarak adlandırılabilir (Kirsch, Chang, & Miller, 1999).

Bir arama sistemi için indeks yapısı oluşturulmaya başlanmadan önce, soyut bir sıralama modeli düşünülmelidir. Belgeler, bilgisayarların doğrudan analiz etmesi zor olan doğal insan dillerinde yazılmıştır. Dolayısıyla, metinler indeks terimlerine dönüştürülürler. Sıralı bir sonuç listesi oluşturmak için, belgeler en yüksek puan alan belgelerin ilk sırada yer almasını sağlayacak şekilde skorlara göre sıralanırlar. Bir kullanıcı sorgusuna karşılık olarak uygun bir sırayla belgeleri sıralamaya çalışan bir insan olduğu varsayılırsa, bu belgeleri "iyi", "pekiyi değil" ve "kötü" gibi yığınlara yerleştirir. Bilgisayarlar da benzer bir puanlama sistemi ile aynı şeyi yapmaktadırlar. Ne yazık ki, belgelerin derin anlamını bulmak bilgisayarlar açısından oldukça zor olduğundan, arama motorları puanlama algoritmalarını kullanarak bu sonuçları elde etmeye çalışırlar.

Belgeler sorguların bulunabilmesini sağlayacak bazı özelliklere sahiptirler. Örneğin, belgelere gelen bağlantı sayıları belgelerin kalitesini ortaya koyarlar. Ayrıca belgelerin güncellenme sayısı da sorgu tipine göre önem teşkil eden bir özelliktir. Mesela "İstanbul da hava durumu" sorgusunu ele alalım. Bu tarz bir sorguda belgenin güncelleme adedine göre sıralama yapılması kullanıcının aradığı belgeyi bulabilmesi açısından büyük bir önem teşkil eder. Bunun tersi olarak da "Çalıkuşu" gibi bir kitap özeti aranmışsa ters mantık olarak belgenin güncellenme sayısı açısından düşük güncellenme sayısına sahip belgenin ilk sıralarda çıkması beklenir.

Arama motorları belgelerin özelliklerini irdeleyerek arama işlemi gerçekleştireceklerinden ve Web içerisinde milyarlarca belge olmasından ötürü bu işlemleri pratik ve hızlı bir şekilde yapmaları gerekmektedir. Bundan dolayı ters indeks yapılarına ihtiyaç duyulur.

#### 3.2.7.1 Ters İndeksler

Tüm modern arama motoru dizinleri ters indekslere dayanmaktadır. Geçmişte diğer indeks yapıları kullanılmış fakat ters indeksler en verimli ve en esnek indeks yapısı olarak kabul edilmişlerdir. Ters indeks yapısı, bir kitabın arkasında bulunan indeks yapısının eşdeğeridir. Kitap indeksi, indekslerine göre alfabetik olarak düzenlenmiştir. Her indeks terimi bu kelimenin geçtiği sayfaların bir listesini içerir. Benzer olarak arama motorlarında da benzer bir yapı kullanılarak, belgelerin kelimelerin bir parçası olarak algılanması sağlanır. Arama motorlarında, indeks teriminin belirli bir belgeye ya da bir yere işaret eden kısmı işaretçi olarak adlandırılır. Koleksiyondaki her bir belgeye, depolamayı verimli hale getirmek için benzersiz bir numara verilir. Ters indeksin en büyük avantajı, sorgunun işlenmesi için indeks dizisinin yalnızca küçük bir bölümüne bakılıp işlem yapılmasıdır (Mahapatra & Biswas, 2011).

#### 3.2.7.2 Belgeler

Tersine çevrilmiş indeks yapısında, indekste bulunan her kelime, bulunduğu belgelerin bir listesini içerir. Şekil 3.5'te, aşağıda gösterilen iki dokümandan oluşturulan bir tersine indeks yapısı gösterilmektedir. İndeks her iki cümlede bulunan her kelimeyi içerir. Her kelimenin yanında, bir kutu listesi vardır ve bu kutuların her biri ait olduğu dokümanı bildirir.

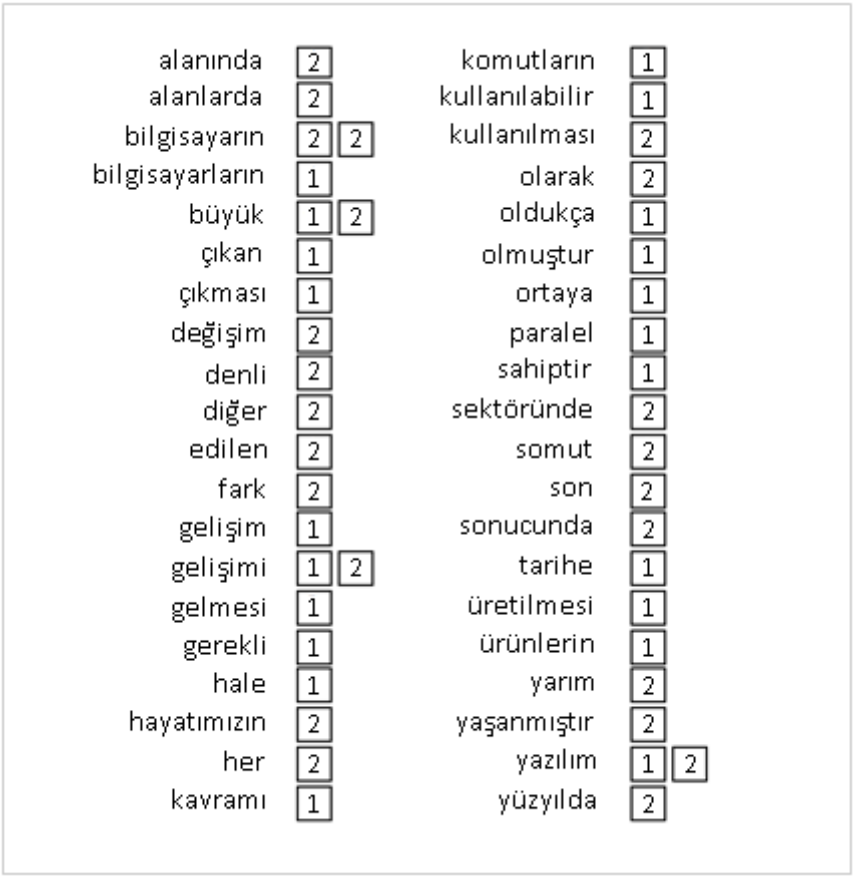
Örneğin; "yazılım" kelimesi 1. ve 2. doküman da bulunan bir kelimedir. Belgelerdeki her terim bir boyuta karşılık gelir, dolayısıyla on binlerce boyut vardır. Bu şekilde indeksin, kelimelerin frekans değerlerinin tutulmadığı görülür. Sadece her kelimenin göründüğü belgeleri kaydedilmiştir. Örneğin, doküman 1 "ortaya" kelimesi iki kez içerirken, doküman 2 "ortaya" kelimesini içermemektedir.

***doküman 1***

Bilgisayarların ortaya çıkması ve gelişimi, bu ürünlerin kullanılabilir hale gelmesi için gerekli komutların üretilmesi ile ortaya çıkan "yazılım" kavramı, bilgisayarların gelişimi ile paralel bir tarihe sahiptir. Bu gelişim oldukça büyük olmuştur.

***doküman 2***

Son yarım yüzyılda bilgisayarın bu denli gelişimi ile yazılım sektöründe ve bilgisayarların diğer alanlarda kullanılması sonucunda hayatımızın her alanında somut olarak fark edilen büyük bir değişim yaşanmıştır.




**Şekil 3.5** Doküman'ların tersine İndeks Yapısı

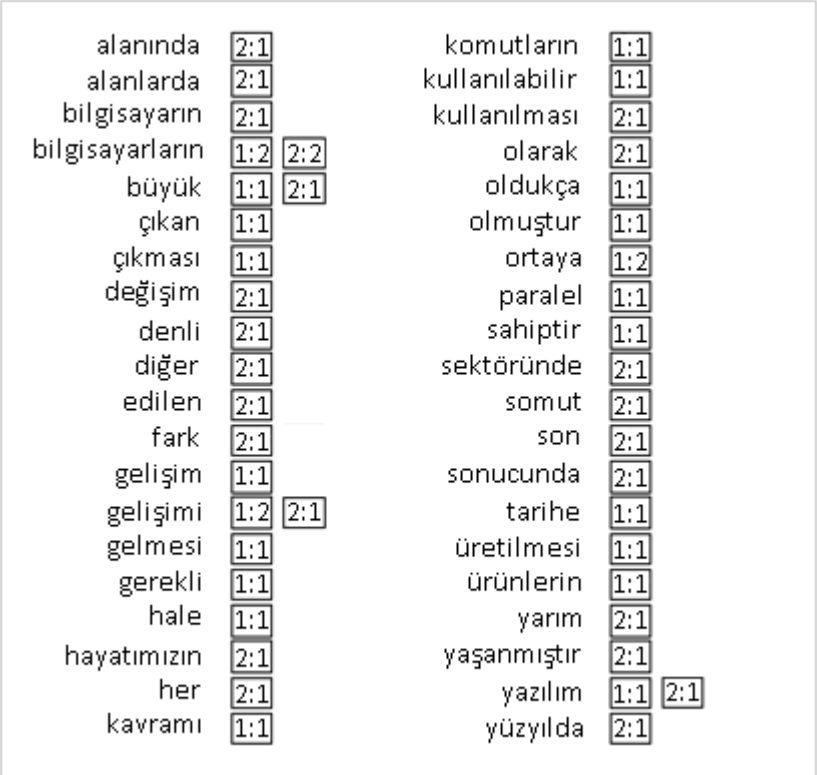
Ters indeksler, kesişim içeren sorgular ile kullanıldığında da sorguların oldukça hızlı çalışmasını sağlayan bir yapıya sahiptirler. Örnek vermek gerekirse; "bilgisayarların" ve "gelişimi" kelimelerini içeren cümlelerin bulunmak istenildiği varsayılırsa; ters indeks yapısı doküman 1 ve 2'de "bilgisayarların" ve "gelişimi"

kelimelerinin olduğunu hızlı bir şekilde gösterir. Her liste cümle numarasına göre sıralandığından, bu indekslerin kesişim noktasını bulmak $O(\max(m, n))$ zaman alır, m ve n bu iki dokümanın uzunluğunu belirtir.

### 3.2.7.3 Frekans Değerleri

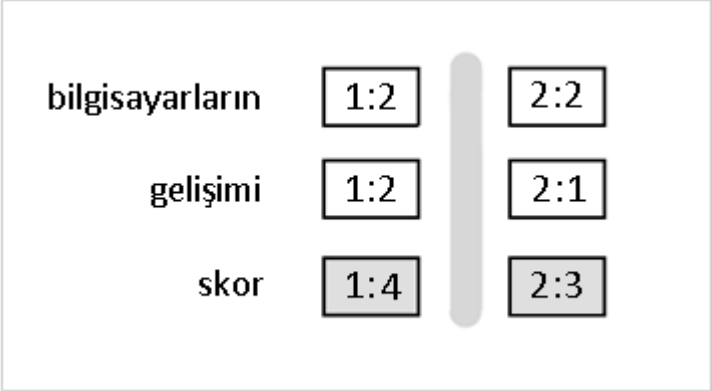
Ters indeks ile dizindeki her kelime bir belge özelliğine karşılık gelmektedir. Bu özellikler belgelerin sıralanması açısından skor değerleri olarak belirlenirler. Yalnızca belge bilgisini içeren bir tersine çevrilmiş indeks, belge açısından tek bir özelliğe sahiptir. Bu özellik aranılan terimin doküman içerisinde var olup olmaması durumudur. Bu önemli bir bilgidir, ancak çok sayıda belge bu terime sahip olduğunda bu belgelerin sıralanması için ekstra bir özelliğe ihtiyaç duyulur (Salton & Buckley, 1988) .

Örneğin, "bilgisayarların gelişimi" sorgusu dikkate alınacak olursa, iki doküman da bu sorguyla eşleşmektedir. Bu çıkarılan sonuç listesinde bu dokümanlardan herhangi birinin diğerine tercih edilmesi için hiçbir neden sunulmamaktadır. Şekil 3.6'daki ters indeks yapısında bir öncekine benzeyen bir ters indeks yapısı vardır.




**Şekil 3.6** Dokümanların Frekans Değerleriyle Oluşturulmuş Ters İndeks Yapısı

Yine aynı kelimeler ve kelimelerin hangi dokümanlarda tutulduğu bilgisi yer almaktadır. Tek fark listelerin ikinci bir sayıya sahip olmasıdır. Bu ikinci sayı, belgede sözcüğün görüntülenme sayısı yani frekans değeridir.




**Şekil 3.7** "bilgisayarların gelişimi" sorgusu sonucu

Bu küçük miktardaki ek veri "bilgisayarların gelişimi" sorgusu için 2. doküman yerine 1. dokümanın seçilmesini sağlar. Çünkü frekans değerleri toplamı açısından en yüksek değere 1. dokümanın sahip olduğu anlaşılmaktadır. Şekil 3.7'de bu örnek gösterilmektedir. Bu örnekte, 1. dokümanın 2. dokümandan içerik açısından daha iyi olduğu tam olarak anlaşılamamaktadır. Ancak genel olarak, kelime sayıları, belge ile ilgili alaka düzeyinin güçlü bir göstergesidirler. Kelime sayıları, belirli bir konuyla ilgili belgelerin bulunmasında yardımcı olurlar. Örneğin, tropikal balıklar ve tropikal adalar hakkında iki belge incelenirse, tropik adalarla ilgili belge muhtemelen "balık" kelimesini içerecektir, ancak "balık" kelimesini ancak birkaç kez içermesi beklenir. Öte yandan, tropikal balıklarla ilgili belge, "balık" kelimesini birçok kez içerecektir. Bu yüzden kelimelerin belgedeki frekanslarının kullanılması, ilgili belgelerin sıralanmasında önemli bir etkendir.

### 3.2.7.4 İndeksleme Algoritması

Arama motorları kendi içlerinde sorgulama yaparken, öncesinde oluşturulmuş indeksler üzerinde sorgulama yaparlar. Basit bir indeksleme algoritmasının Pseudo kodu Şekil 3.8'de gösterilmektedir. Süreç sadece birkaç adımdan ibarettir. Belgelerin listesi BuildIndex işlevine aktarılır ve işlev her belgeyi Token'lara ayrıştırır. Bu Token'lar noktalama işaretlerinden ve bazı eklerden temizlenir ve kökleri bulunur.

Ardından bir tablo kullanılarak yinelenen Token'lar kaldırılır. Daha sonra, her bir token için, indekste yeni bir ters çevrilmiş listenin oluşturulup oluşturulmadığı kontrol edilir ve gerekirse bir tane oluşturulur. Son olarak, geçerli belge numarası frekansı ile birlikte ters çevrilmiş listeye eklenir. Sonuç, bir Token tablosu ve ters listeden oluşur. Tersine çevrilmiş listeler, sadece belge numaralarının listesidir ve özel bilgiler içermezler (Larson, 2010).

```
function BuildIndex(documents)
        foreach documents do
                tokens =Parse(documents[n])
                foreach tokens.next() do
                        token = tokens.getToken()
                        url = tokens.getUrl()
                        frequency = tokens.getFrequency()
                        inverseIndex.add(token, url, frequency )
                end for
        end for
end function
```

**Şekil 3.8** İndeks algoritması için Pseudo kod yapısı

## 3.3 Sorgulama Mimarisi

### 3.3.1 Genel Bakış

Bu bölümde arama motoru sorgularının bir bilgisayar tarafından nasıl işlendiği anlatılmaktadır, bu nedenle bu bölümün tamamı sorgulama işlemi olarak adlandırılabilir. Verimli sorgulama işlemi Web arama motorlarında önemli bir sorundur (Liu, 2009). Dünyanın her yerinden insanlar, her gün yarım milyardan fazla sorgu ile milyarlarca web sayfası içerisinden arama yapmaktadırlar.

### 3.3.2 Sorgu Algoritması

Bir ters indeks yapısı oluşturulduktan sonra, sorgu sonuçlarını üretmek için verilerin işlenmesi gerekir. Şekil 3.9'de her bir sütun farklı bir dokümanı ve her bir satır ters çevrilmiş indeksi temsil etmektedir. Bu örnekteki tersine çevrilmiş indeksler kelime

frekanslarını tutar ve skor, her bir belgedeki kelime frekanslarının toplamıdır. İlk adımda, ilk belge için tüm frekanslar, bu belgenin puanını üretmek için toplanır. İlk belge için puanlama tamamlandıktan sonra, ikinci belge puanlanır. Tek kelime içeren sorgu algoritmasının pseudo kod yapısı aşağıda gösterilmektedir. Bu algoritmada sırasıyla; "Q" sorguyu, "InvertedList" indeksi, "documentCollection" elde edilen doküman koleksiyonunu göstermektedir. Sorgudaki her kelime için, o kelimeye ait ters indeks yapısı getirilir. Bu ters listedeki her elemanın gösterdiği dokümana ve frekansına erişilerek documentCollection listesine, dokümanın frekans değeri ve pagerank değeri toplanarak dokümanla birlikte eklenir. Ardından oluşturulan documentCollection listesi elde edilen skor değerlerine göre sıralanır ve sonuç elde edilir.

```
function DocumentRetrieval(Q)
        InvertedList, documentCollection(document,score)
        document
        Q = trim(Q)

        If not Q.Contain(" ") then
                InvertedList = getInvertedList(Q)
        end if

        foreach item in InvertedList do
                document = getDocumentById(item.docId)
                documentCollection.add(document,document[pageRank] + item.Frequency)
        end for

        documentCollection  = sortByFrequency(documentCollection)
end function
```

**Şekil 3.9** Sorgu algoritmasının tek kelime arama yaparken kullanılan Pseudo kod yapısı.

### 3.3.3 Birleşik Sorgulama

Birleşik sorgulama, kullanıcıya gönderilen her belgenin tüm sorgu terimlerini içermesi gerektiği anlamına gelmektedir. Dünyada popüler olarak kullanılan web arama motorları birleşik sorgulama yaparak çalışırlar. Birleşik sorgulamada, kısa sorgular etkinliği ve verimliliği arttırırken, paragraf tarzındaki uzun sorgular arama motorlarına yüksek bir işlem yükü bindirirler. Şekil 3.10'da birleşik sorgulamanın pseudo kod yapısı gösterilmektedir. Birleşik sorgulamada aynı yöntem kullanılarak sorgulama yapılırken, ekstra olarak her kelime için bir sorgulama yapılıp doküman koleksiyonuna aktarılır,

ardından koleksiyonun içinde bulunan aynı dökümanlar koleksiyondan çıkartılır. Ardından sonuçlar puanlarına göre sıralanıp sonuç listesi döndürülür.

```
function DocumentRetrieval(Q)
        InvertedList, documentCollection(document,score)
        resultCollection(document,score)
        Q = trim(Q)
        List<String> tokenList = Q.Split(" ")

        foreach token in tokenList do

                InvertedList = getInvertedList(Q)

                foreach item in InvertedList do
                        document = getDocumentById(item.docId)
                        score = document[pageRank] + item.Frequency
                        documentCollection.add(document, score)
                end for

        end for

        while not documentCollection.empty() do
                document = documentCollection.pop()
                score = documentCollection.getScore(document)
                while documenCollection.Exist(document.docId) do
                        doc = documenCollection.getAndDeleteDocument(document.docId)
                        score +=  documentCollection.getScore(doc)
                end for
                resultCollection(document,score)
        end for

end function
```

**Şekil 3.10** Sorgu algoritmasında birden çok kelime araması yaparken kullanılan Pseudo kod yapısı.

### 3.3.4 Dağıtık Mimari

Modern bilgisayarlar büyük işlem yüklerini kaldırabilecek kapasitededirler ve hatta günümüzde kullanılan çoğu görev için yeterli olabilirler. Fakat büyük bir kuruluş için çok sayıda kullanıcıyla çalışmak, birden fazla bilgisayar kullanmayı gerektirebilir. Birden fazla bilgisayar kullanımındaki temel yaklaşım, tüm sorguların yönetici bir bilgisayara gönderilmesidir.

Yönetici bilgisayarlar, işlevleri gereği, sorgu işlemlerinin bir kısmını işler ve sorgu işlemenin geri kalan kısmını yönettiği indeks sunucularına gönderirler. İndeks sunucuları ise bu sorguları işleyip ve gerekli sonuçarı üretip yönetici bilgisayara

sonuçları gönderirler. Yönetici bilgisayarlar da gelen sonuçları düzenlerler ve düzenlenen bu sonuçları kullanıcılara geri gönderirler (Croft et al., 2010).

Dünyada en kolay ve en çok kullanılan dağıtık mimari stratejisi belge dağıtımıdır. Bu stratejide, her indeks sunucusu bir arama motoru görevi görür. Yönetici bilgisayar, bütün indeks sunucularına, sorgunun bir kopyasını gönderir ve her indeks sunucusu, sonuçların belge puanları dâhil olmak üzere, sonuçları sıralanmış bir şekilde, yönetici bilgisayara geri döndürür (Borthakur, 2007). Bu sonuçlar, yönetici bilgisayar tarafından tek bir sıralı liste halinde birleştirilir ve sonuçlar kullanıcıya döndürülür.

Terim dağıtımı olarak da bilinen bir dağıtık mimari stratejisi daha vardır. Terim dağıtımında, aynı belge dağıtımında olduğu gibi birçok indeks sunucusu bulunmaktadır. Fakat burda en büyük fark, indeks sunucuları, yönetici bilgisayar tarafıdan gönderilen sorguyu işlemek yerine, ilgili terime ait en uzun ters indeks yapısını tutan indeks sunucularından biri, sorguyu işlemek için seçilir. Diğer indeks sunucularında ilgili terime ait ters indeks yapısı varsa, bu veriler ağ üzerinden sorguyu işleyen indeks sunucusuna gönderilir. Sorgu işleme tamamlandığında, sonuçlar yönetici bilgisayara gönderilir. Bu mimaride indeks sunucuları arasında, ters indeks verilerinin birbirlerine gönderilmesi gerektiğinden, doküman dağıtımından daha karmaşık bir yapı vardır. Ayrıca bu mimaride, ters indeks yapılarının boyutu göz önünde bulundurulduğunda, indeks sunucularının bu verileri birbirlerine göndermesi sonucu ağın şişmesi ve yavaşlaması durumu gerçekleşebilmektedir. Buna ek olarak, her sorgu, bütün bilgisayarlar yerine yalnızca bir bilgisayar kullanılarak işlenmektedir. Bir k sayıda terim içeren sorgu olduğu düşünülürse ve n tane indeks sunucusu var ise, bir sorguyu işlemek için gerekli toplam zaman; terim ile dağıtılmış bir sistem için O(kn) iken belge ile dağıtılmış bir sistemde sadece O(k) olur (Croft et al., 2010).

### 3.3.5 Önbelleğe Alma

Kullanıcıların yapmış oldukları sorgular genellikle benzer sorgulardır. Bir arama motorunun her gün aldığı sorguların yaklaşık yarısı daha önceden yapılmış sorgulardır. Bu nedenden ötürü arama motorları önbellek sistemi kullanırlar. Genel olarak, önbelleğe alma, daha sonra kullanmak isteyebileceğiniz bir şeyi saklama anlamına gelmektedir. Önbellek alma arama motorları için yüksek uygunlukta bir sistemdir. Sorgular ve sıralanmış listelerin boyutları oldukça küçüktür. Bu yüzden yapılan sorguların ve bu sorguların sonuçlarını bir önbellekte saklamak için disk üzerinde fazla

alana ihtiyaç duyulmaz. Buna karşılık, bir sorgunun işlenmesi birçok hesaplama ve zaman gerektirmektedir. Bu yönden bakıldığında web arama motorları için yapılan sorguların sonuçlarıyla birlikte ayrı bir tabloda tutulması, arama motorlarının hızları açısından oldukça büyük bir anlam ifade etmektedir. Ancak önbellekleme, arama motorlarının tüm performans sorunlarını çözmez, çünkü, her gün alınan sorguların yaklaşık yarısı benzersiz olmaktadır (Baeza-Yates et al., 2008).

### 3.3.6 Bölgesel Arama

Arama motorlarının kaliteli sonuçlar üretmesinde kullanılan bir diğer etkili yöntem ise bölgesel arama sisteminin arama motorlarına entegre edilmesidir. Bu sistem ile arama sonuçlarının sıralamasını değiştirmek için sorguyu gönderen kullanıcının bilgisayarından veya mobil cihazından elde edilen coğrafi bilgiler kullanılarak bölgesel arama işlemi gerçekleştirilmektedir. Örneğin, "balıkçılık malzemeleri" sorgusu, ülkenin (veya dünyanın) her yerinden balıkçılık malzemeleri tedarikçilerini içeren uzun bir sonuç listesi oluşturulacaktır. Fakat bu sistem kullanıldığında, "balıkçılık malzemeleri" ve konum bilgisi ile yapılan bir aramada, arama motoru konum bilgisi sayesinde, verilen konum bölgesindeki tedarikçilerin web sayfalarını, diğer tedarikçilerin web sayfalarından daha üst sıralara koyabilmektedir. Bölgesel arama desteğine sahip arama motorları aşağıdaki adımları kullanarak çalışmaktadırlar. (Baker et al., 2011)

1. Web sayfaları toplanırken, belgeye eklenen konum meta verileri veya belge metninde bulunan yer adları, şehir adları veya ülke adları saptanır ve ayrı bir özellik olarak belge ile birlikte tutulurlar.

2. Kullanıcılardan sorgu bilgisi alınırken ek olarak kullanıcıların bilgisayarları da konum bilgisi sağlıyorsa, bilgisayarların konum bilgisi elde edilerek sorgu işlemesi yapılır.

3.Sorgu sonucunda belgeler sıralanırken, kullanıcıdan alınan konum bilgisi kullanılarak, kullanıcıya yakın konumdaki belgelerin puanlarında ekstra puan artışı yapılır. Bu sayede kullanıcıya yakın olan içerikler sonuç listesinde üst sıraya çıkartılmış olur.

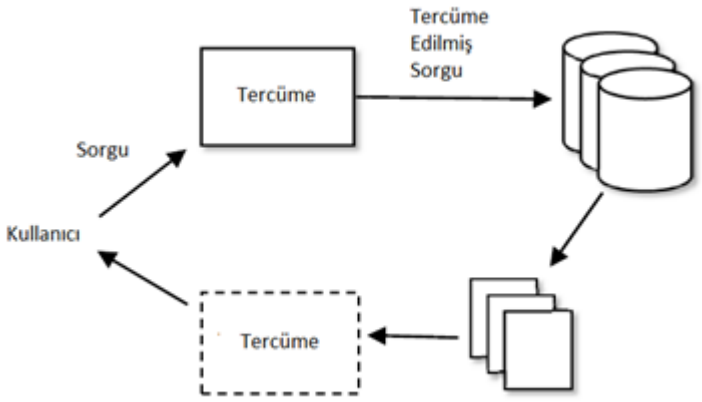
### 3.3.7 Sonuçların Gösterilmesi

Bir arama motorunun başarılı görünmesini ve popüler olmasını sağlayan en önemli faktör, kullanıcıların arama motorunun ürettiği sonuçları anlamasıdır. Arama çıktısının görüntülenmesi için pek çok görselleştirme tekniği önerilmiştir (Hearst, 1999) ancak çoğu arama motoru için sonuç sayfaları, web sayfalarına bağlı belge özeti bilgisinden oluşmaktadır. Web arama motorları, sonuç listesinin her bir elemanı için, web sayfasının başlığını, URL'sini, sayfanın önbelleğe alınmış sürümünü ve kısa bir metin özetini gösterirler. Ayrıca sonuç sayfalarının çoğu, kısa açıklamalardan ve bağlantılardan oluşan reklamlar da içerirler. Başlık, URL ve sorgu kelimeleri, genellikle kalın yazı tipinde gösterilerek vurgulanır. Şekil 3.11'de bir web arama motorunun sonuç sayfasından alınan bir belge özeti örneği gösterilmektedir.


**Şekil 3.11** Örnek bir arama sonucu

### 3.3.8 Farklı Dillerde Yapılan Sorgular

Farklı dillerde yapılan sorguları çalıştıran arama motorları, sorguları tercüme ederek ve çapraz dil araması yaparak sonuçları listelerler. (bkz. Şekil 3.12).



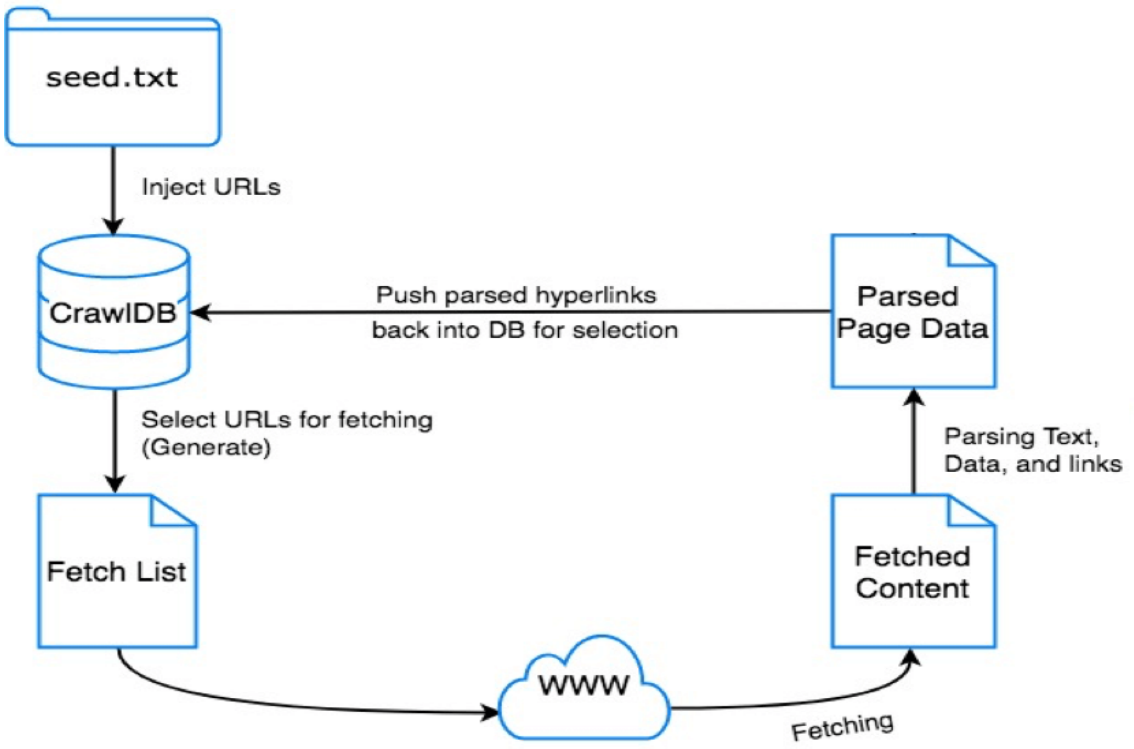

**Şekil 3.12** Çapraz dil araması

Şekilde gösterildiği üzere arama motoru sorguyu bir dilde alır (ör. Türkçe). Ardından sorguyu kendi içerisinde kullandığı dil yapısına göre tercüme eder ve tercüme edilmiş sorgu ile yeni sonuçlar üretir. Üretilen sonuçlar arama motorunun kullandığı dilde olduğundan, belgeler kullanıcının kullanmış olduğu Fransızca, Türkçe veya Çince gibi dile çevrilip, sonuçlar çevrilmiş bu dil üzerinden kullanıcıya gösterilirler (Maeda et al., 2000).

### 3.4 Apache Nutch

Nutch projesi, 2002 yılında Doug Cutting kişi tarafından geliştirilimiş açık kaynak kodlu bir Web Crawler projesidir. Nutch, açık kaynak kodlu bir arama motoru geliştirme düşüncesiyle oluşturulmuştur. Nutch, Java diliyle yazılmış ve daha çok linux sistemleriyle uyumlu çalışabilen bir Web Crawler programıdır (Anonim, 2018c).

Nutch, verilen kök linkler ile verilen derinlikte web sayfalarını kaydeden ve paralel olarak çalışabilen bir Web Crawler uygulamasıdır. Nutch projesinin çalışma mimarisi Şekil 3.13'te gösterilmektedir.

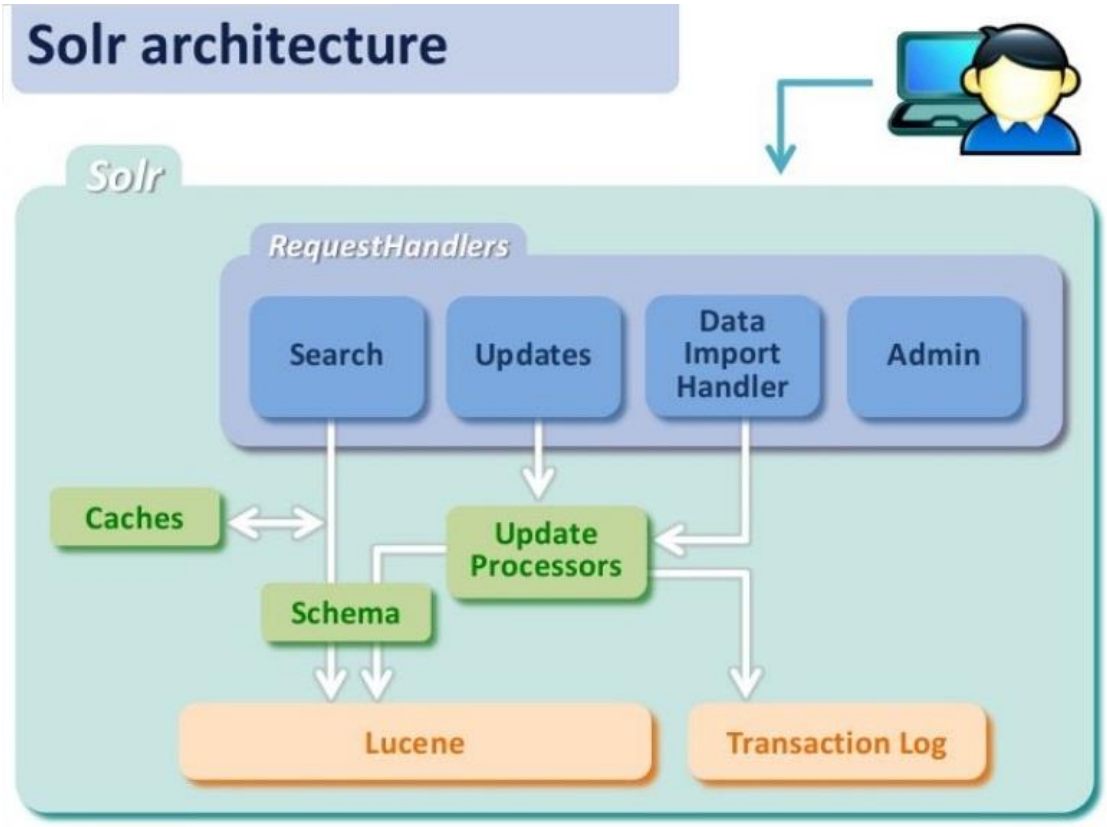



**Şekil 3.13** Apache Nutch Çalışma Mimarisi (Anonim, 2017b)

### 3.5 Apache Solr

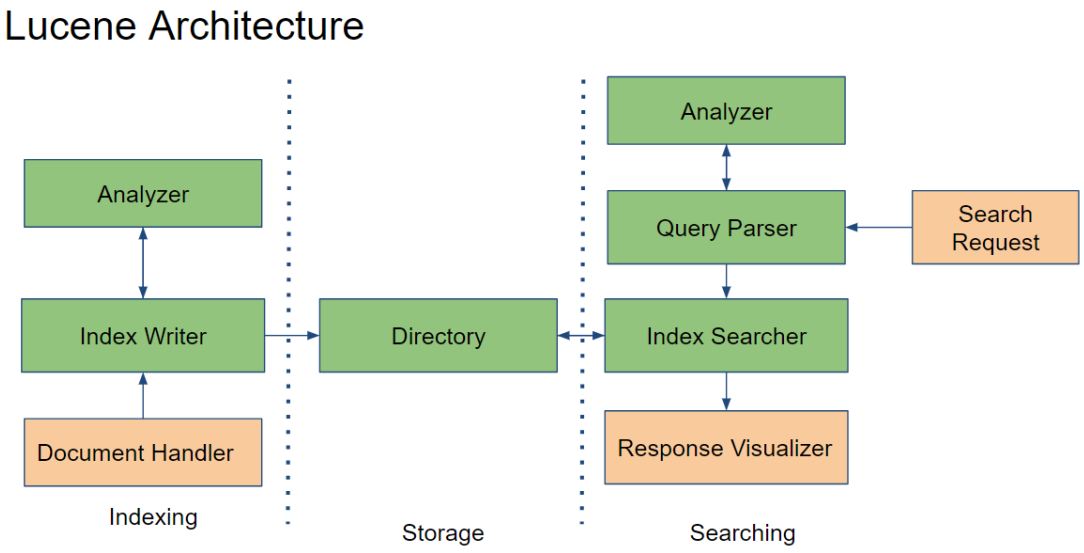
Solr, Apache Lucene arama kütüphanesi üzerine kurulu Java ile geliştirilmiş, arama konusundaki yetenekleri ile ön plana çıkan açık kaynak kodlu, Tomcat ve Jetty gibi sunucular üzerinde çalıştırılabilen bir Apache ürünüdür (Yeşilkaya, 2013) .

Solr, Lucene temelini kullanan filtreleme,önbellekleme ve sorgulama işlemleri yapan, Microsoft Office belgelerinde veya pdf benzeri belgelerde indeksleme yapabilen, HTTP/XML veya JSON gibi bir çok programlama dili ile desteklenen protokollerle çalışabilen açık kaynak kodlu indeksleme ve sorgulama kütüphanesidir. Solr kütüphanesinin çalışma mimarisi Şekil 3.14'te gösterilmektedir. Lucene tabanlı arama motorlarının bir ailesi olan Apache Solr, Lucene kütüphanelerini kullanarak Tomcat, Jetty gibi sunucularda servlet olarak çalışır (Bulut, 2014).




**Şekil 3.14** Apache Solr Çalışma Mimarisi (Manios, 2015)

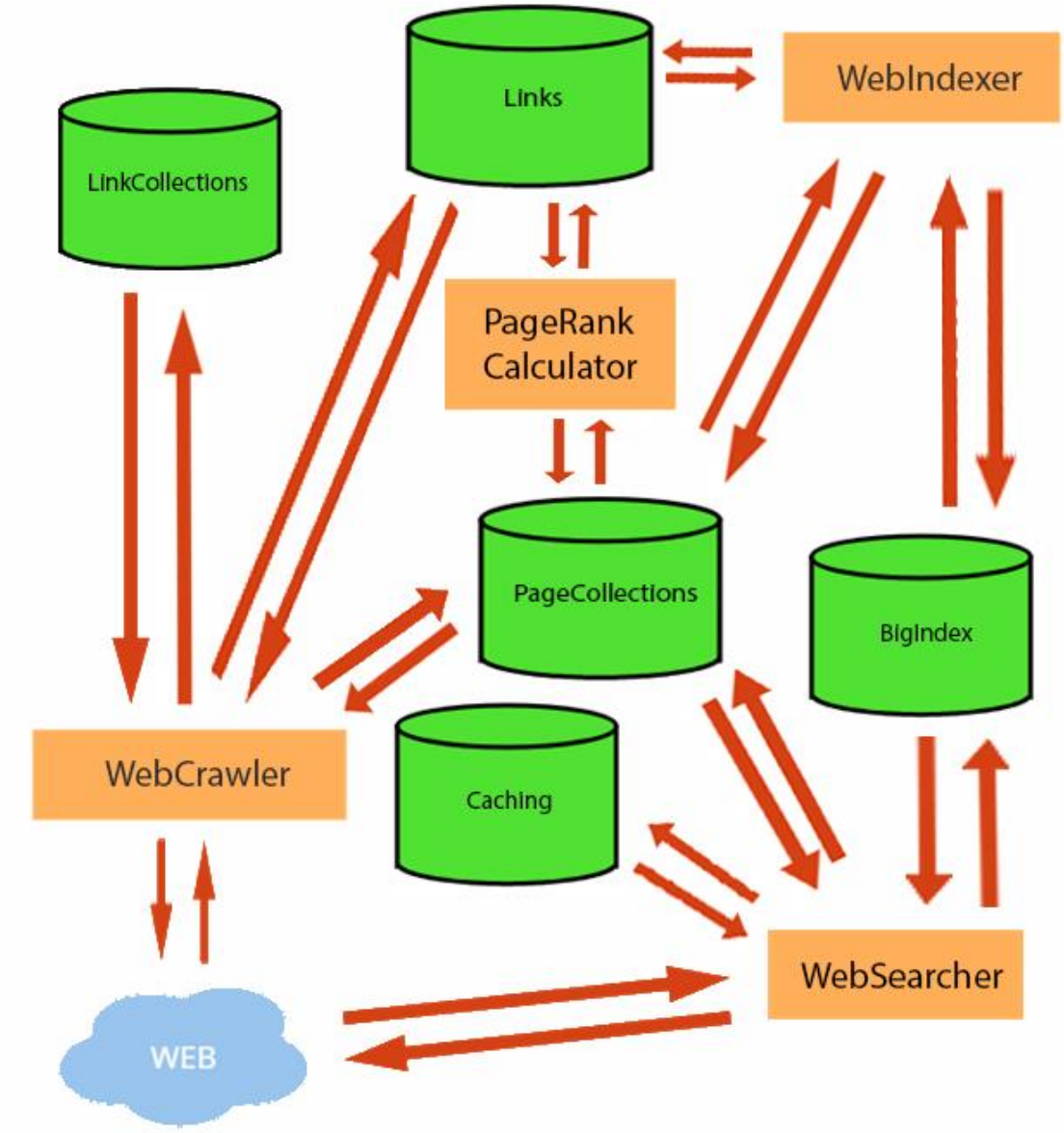
Lucene açık kaynak kodlu metin indeksleme ve arama kütüphanesidir. Solr ve Lucene arasındaki ilişki araba ve araba motoru arasındaki ilişkiye benzetilmektedir. Apache Lucene çalışma mimarisi Şekil 3.15'te gösterilmektedir.




**Şekil 3.15** Apache Lucene Çalışma Mimarisi (Reddy, 2017)

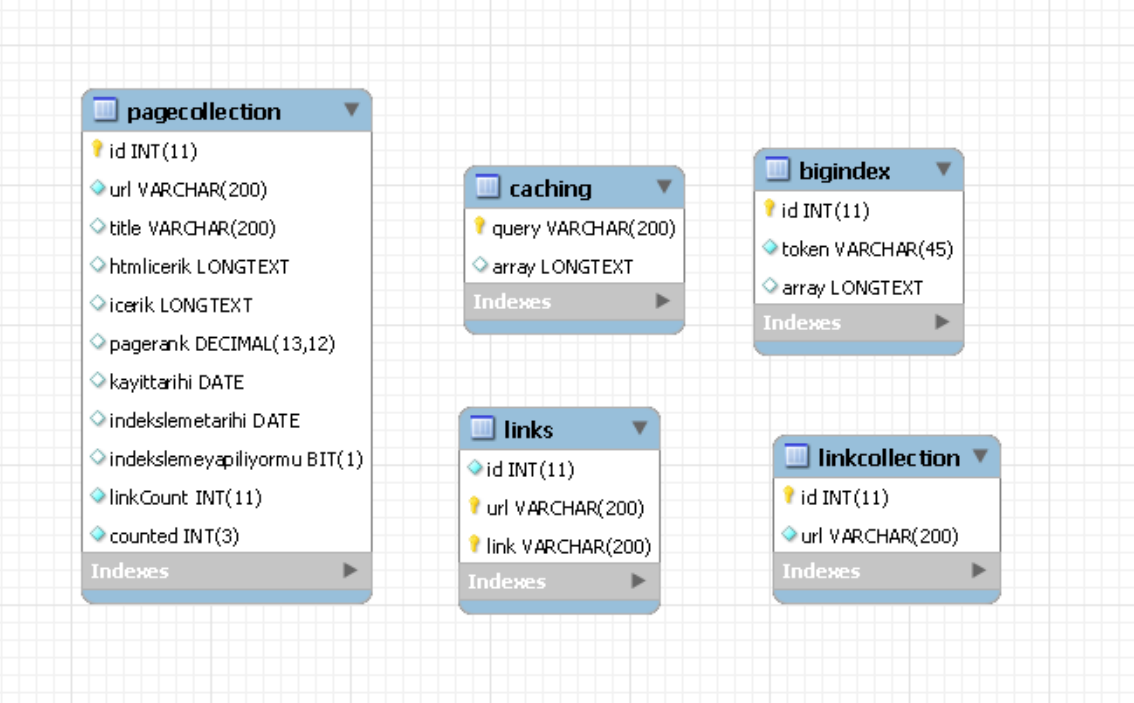
# 4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Yapılan arama motoru uygulaması 4 çekirdekli ve 8 GB ram kapasitesine sahip Apple Macbook Pro bilgisayarı üzerinde çalıştırılmıştır. Arama motoru mimarisi WebCrawler, WebIndexer, WebSearcher ve PageRankCalculator adında dört bileşenden oluşmaktadır. Arama motorunun geliştirilmesinde C# .NET programlama dili kullanılmıştır. Bu tez çalışmasında gerçekleştirilen arama motorunun mimarisi Şekil 4.1 de gösterilmektedir. Ayrıca projenin kodları ve veritabanı kodları github sitesinde "https://github.com/mehmetkarlik/search_engine" adresi altında paylaşılmıştır.




**Şekil 4.1** Arama Motoru Mimarisi

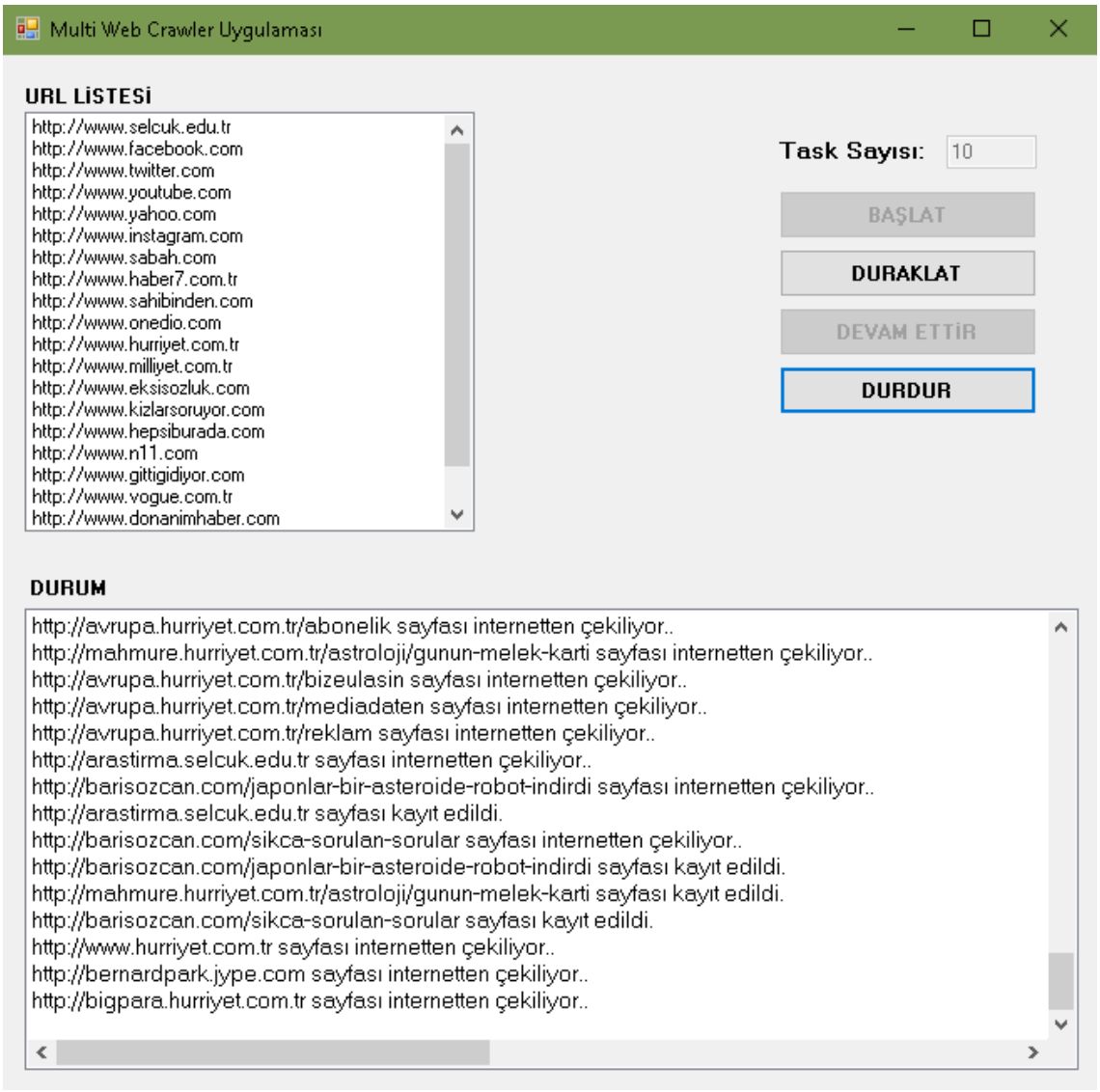
WebCrawler tarafından indirilen web sayfalarını ve linklerini ve WebIndexer'ın web sayfaları üzerinde yaptığı işlemler sonucu oluşturduğu indeks yapısını kaydetmek için MySQL veri tabanı kullanılmıştır. Veri sorgulama, güncelleme, kaydetme ve silme işlemleri için uygulamanın belirli yerlerinde SQL sorgu dili kullanılmıştır. Tezin bu bölümünde, belirtilen dört aşama detaylı olarak anlatılmaktadır. Arama motoru uygulamasının veritabanı içerisinde kullandığı tablolar ve bu tabloların içerisinde yer alan alanların adları Şekil 4.2' de gösterilmektedir.




**Şekil 4.2** Uygulama İçerisinde Kullanılan Veritabanı Tabloları

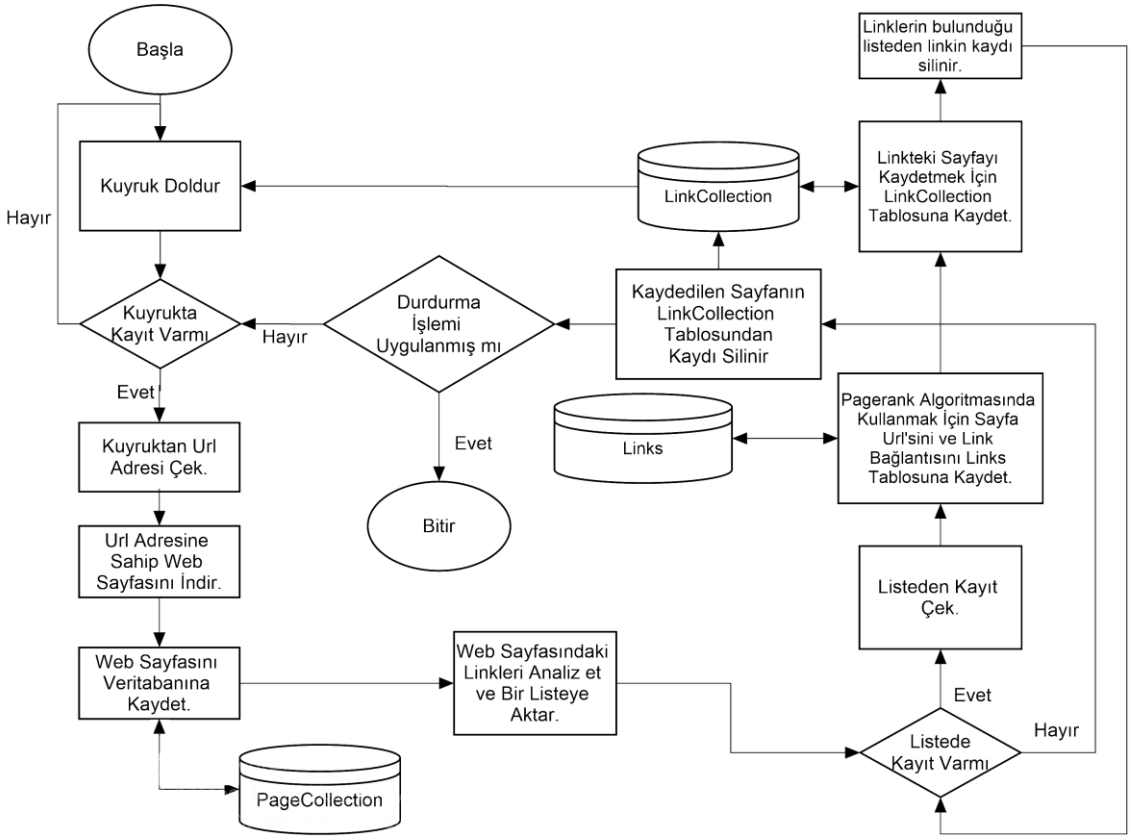
## 4.1 WebCrawler Uygulaması

WebCrawler uygulamasının kullanıcı arayüzü Şekil 4.3 de görülmektedir. Arayüzde görüldüğü üzere başlat, durdur, duraklat, devam ettir butonları sayesinde istenildiği gibi yönetilebilir bir yapıya sahiptir. Formun URL listesi kısmında, daha önceden belirlenmiş olan, Türkiye'de popüler olarak kullanılan web sitelerinin başlangıç adresleri gösterilmektedir. Uygulamanın internetten verimli bir şekilde veri toplayabilmesi için uygulama paralel programlama konseptleri kullanılarak geliştirilmiştir.




**Şekil 4.3** WebCrawler Uygulaması

Uygulama çalıştırıldığında, uygulamanın yüklenme kodu içerinde formda görülen Url listesindeki bütün web adresleri MySQL veritabanında bulunan LinkCollection tablosuna kayıt edilmektedir. Uygulama yüklendikten sonra Başlat butonuna basıldığında, form içerisinde verilen Task sayısı kadar Task oluşturulur ve bu Task'lar eşzamanlı olarak çalışırlar. Oluşturulan bu Task'ların içerisinde crawl() metodu çalışmaktadır. Crawl metodu içerisinde, bu metodun sürekli çalışmasını sağlayan sonsuz döngü yapısı bulunmaktadır. Bu döngü yapısından çıkmak için form'da bulunan DURAKLAT veya DURDUR butonları kullanılmaktadır. WebCrawler programının içerisinde bulunan kodların akış diyagramı Şekil 4.4'te gösterilmektedir.




**Şekil 4.4** WebCrawler Uygulamasının Akış Diyagramı

Şekil 4.4'teki diyagramda görüldüğü üzere WebCrawler uygulaması sırasıyla aşağıdaki adımları izleyerek çalışmaktadır.

1. İlk olarak veritabanındaki LinkCollection tablosundan 1000 adet kayıt çekilir ve paralel yapıda çalışan bir kuyruk yapısına bu kayıtlar aktarılır. Ardından Task nesneleri oluşturulur ve çalıştırılır.
2. Kuyrukta kayıt var mı yok mu kontrol edilir, kayıt yoksa 1. adıma geri dönülür, kayıt varsa bir sonraki adıma geçilir.
3. Kuyrukta kayıt olduğu tespit edildikten sonra, paralel çalışan kuyruk yapısından ilgili kayıt çekilir.
4. Kayıtta bulunan Url adresine sahip web sayfasını indirmek için bir HttpWebRequest nesnesi oluşturulur ve bu nesne yardımı ile web sayfasının kaynak kodları ve başlık (Headers) bilgileri indirilir.
5. İndirilen web sayfasının kaynak kodları, web sayfasının başlık özelliklerinde bulunan "Last-Modified" özelliği ve başlangıç değerli (0.15) bir pagerank değeri ile birlikte veritabanındaki PageCollections tablosuna kayıt edilir.

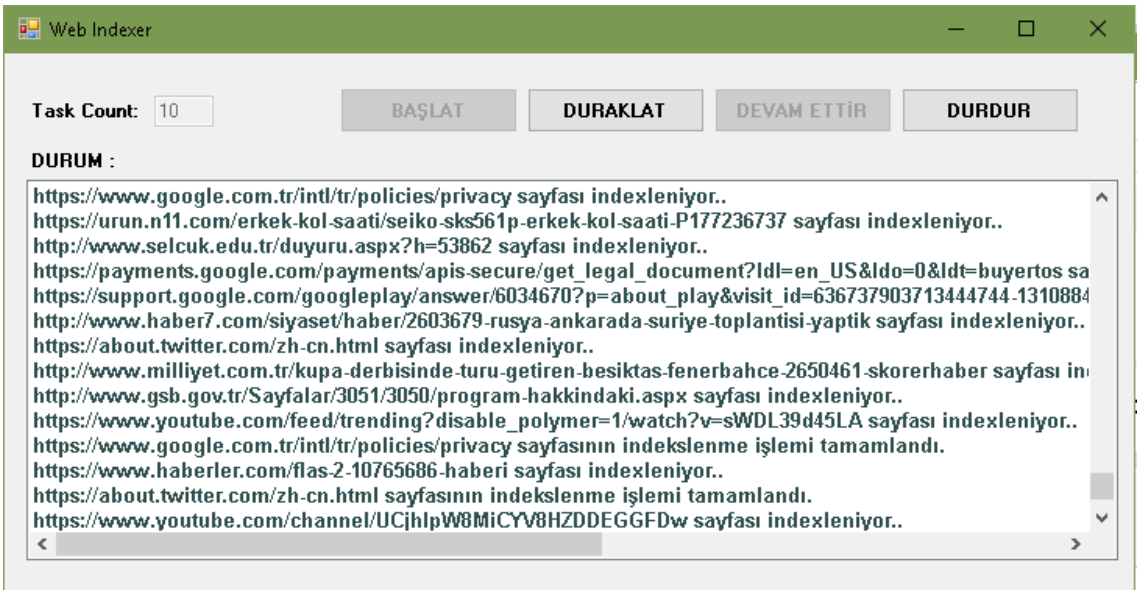
6. Web sayfasındaki linkler analiz edilir ve bir listeye aktarılır.
7. Listede kayıt varmı kontrol edilir, kayıt varsa bir sonraki adıma geçilir, liste boşsa 12. Adımdan devam edilir.
8. Listeden link kaydı çekilir.
9. Pagerank programında kullanılmak üzere kaydedilen web sayfasının Url adresi ile birlikte linkin Url adresi veritabanındaki Links tablosuna kayıt edilir.
10. Çekilen link kaydına ait web sayfasını daha sonra kaydetmek için linkin Url adresi, veritabanındaki LinkCollections tablosuna kaydedilir.
11. Çekilen link kaydı linklerin bulunduğu listeden silinir ve 7. Adıma geri dönülür.
12. Kaydedilen web sayfasının Url adresi LinkCollections tablosundan silinir.
13. Formda durdurma işlemi gerçekleştirildi mi gerçekleştirilmedi mi kontrolü yapılır. Durdurma işlemi gerçekleştirilmediyse 2. adıma gidilir, durdurma işlemi gerçekleştirilmişse algoritma akışından çıkış gerçekleştirilir.

Yukarıdaki sıralamada WebCrawler uygulamasının akış diyagramı detaylı olacak şekilde adım adım anlatılmıştır. Daha detaylı bilgi için Ek A'da bulunan WebCrawler uygulamasının kodları incelenebilir.

### 4.2 WebIndexer Uygulaması

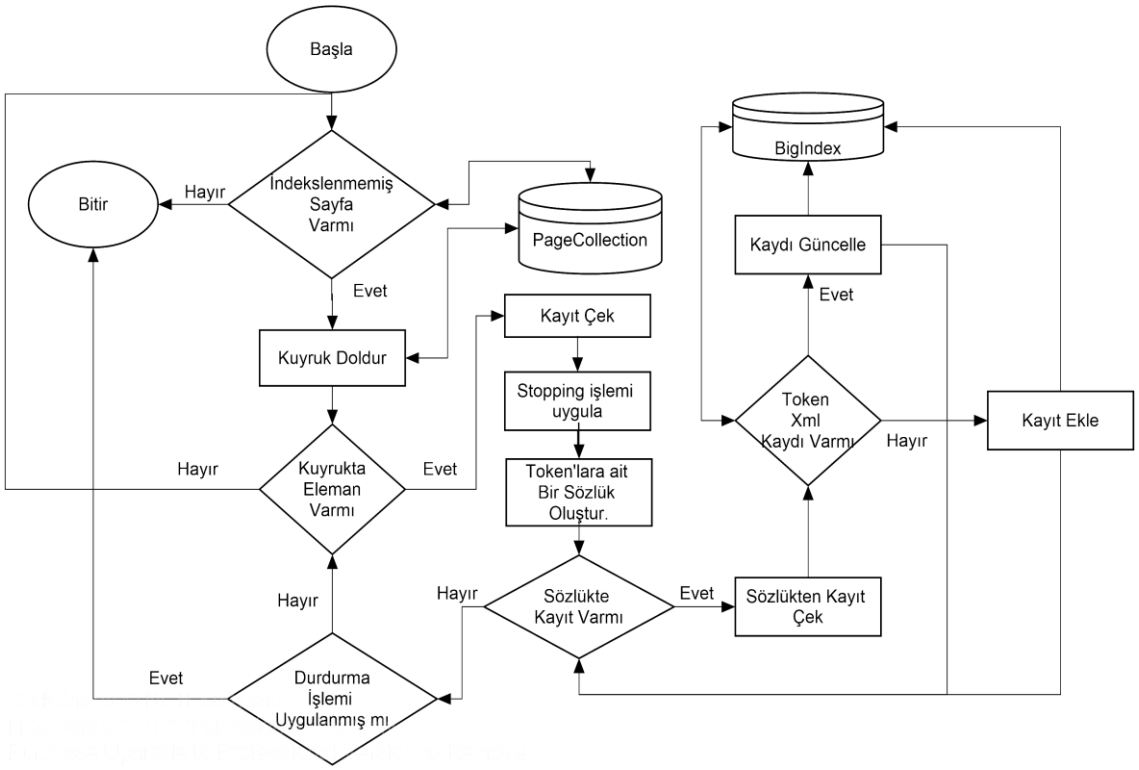
WebIndexer uygulamasının kullanıcı arayüzü Şekil 4.5 de görülmektedir. Program arayüzde görüldüğü üzere aynı WebCrawler uygulamasında olduğu gibi başlat, durdur, duraklat ve devam ettir butonları bulunmaktadır. Bu butonlar sayesinde uygulama istenildiği gibi yönetilebilir bir yapıya sahip olmaktadır.

WebIndexer uygulaması işlemcinn sahip olduğu gücü tam anlamıyla kullanabilmek için paralel programlama konseptlerine uygun olarak geliştirilmiş bir yapıya sahiptir. WebIndexer, WebCrawler ile çalışma mimarisi açısından birbirlerine benzediklerinden, WebIndexer'ın paralel kod mimarisi, WebCrawler'ın paralel kod mimarisi ile aynı özellikleri taşımaktadır. WebIndexer programı, adından da anlaşılacağı üzere, veritabanındaki web sayfalarını indekslemek için geliştirilmiş bir programdır.




**Şekil 4.5** WebIndexer Uygulaması

Bu program sayesinde WebCrawler tarafından veritabanına kaydedilen web sayfaları, indekslenerek, WebSearcher uygulaması tarafından sorgulanabilecek bir yapıya dönüştürülmektedir. Uygulamanın akış diyagramı Şekil 4.6'da gösterilmektedir.



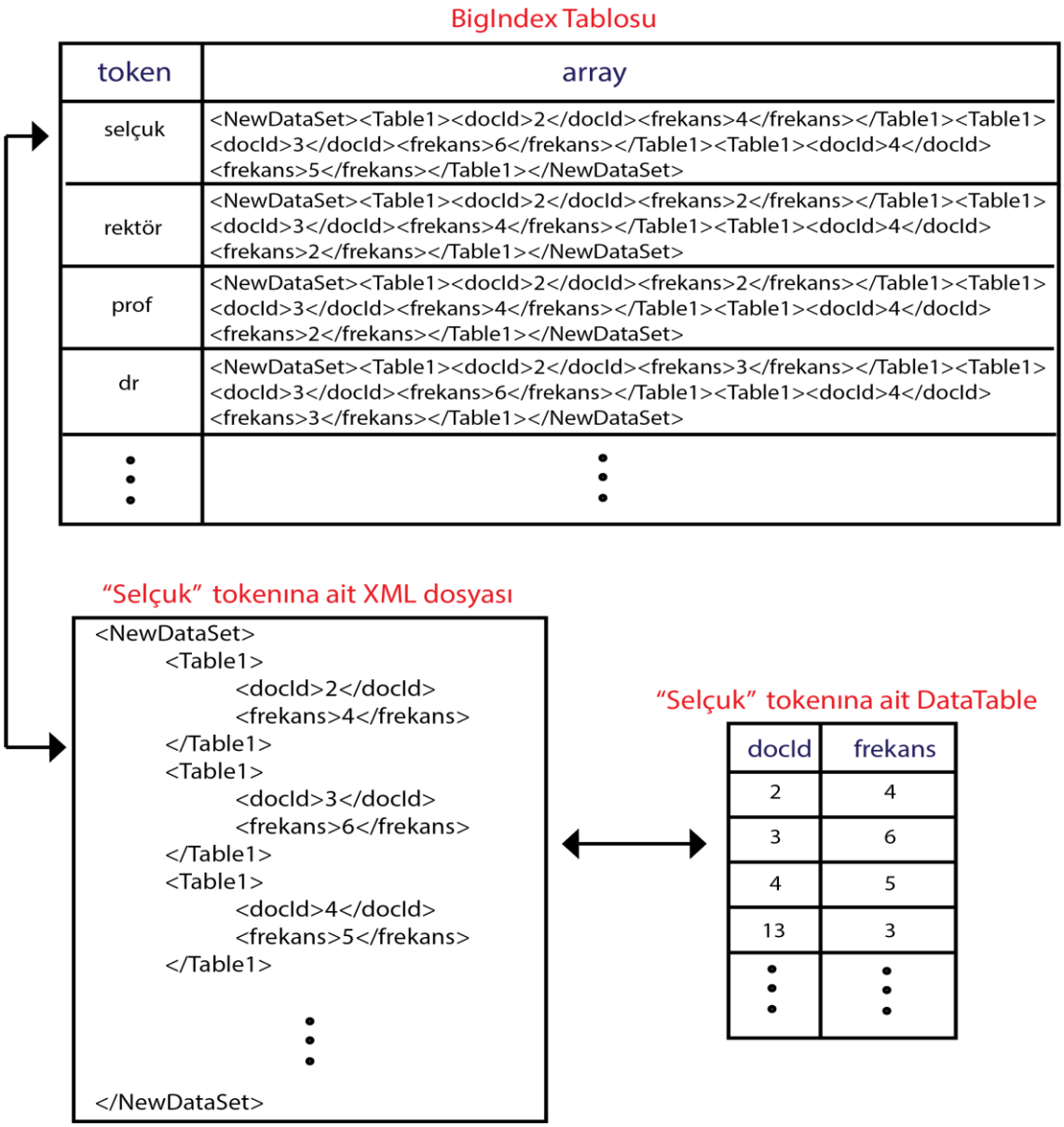

**Şekil 4.6** WebIndexer Uygulamasının Akış Diyagramı

Şekil 4.6'daki diyagramda görüldüğü üzere WebIndexer uygulaması sırasıyla aşağıdaki adımları izleyerek çalışmaktadır.

1. İlk olarak veritabanındaki PageCollections tablosuna bakılarak, indekslenmemiş sayfa var mı yok mu kontrolü yapılır. İndekslenmemiş sayfa yoksa programdan çıkış gerçekleştirilir, indekslenmemiş sayfa varsa bir sonraki adıma gidilir.
2. PageCollections tablosundan indekslenmemiş 1000 adet kayıt çekilerek, paralel yapıya uygun çalışan bir kuyruk yapısına aktarılır.
3. Kuyrukta kayıt kayıt olup olmadığının kontrolü yapılır, kayıt varsa bir sonraki adıma geçilir, kayıt yoksa 1. adıma gidilir.
4. Kuyruktan kayıt çekilir ve bir sonraki adıma gidilir.
5. Kuyruktan çekilen web sitesi kaydının içeriği parçalanarak bir diziye aktarılır, ardından dizideki elemanlar incelenerek Stopping işlemi uygulanır.
6. Kelime dizisindeki kelimeler benzersiz olacak şekilde frekans değerleri hesaplanarak Dictionary<string,int> şeklinde bir sözlük yapısına aktarılır.
7. Sözlük yapısında elemanın olup olmadığı kontrol edilir, eğer eleman varsa bir sonraki adıma geçilir, eleman yoksa 12. adıma gidilir.
8. Sözlük yapısından kayıt çekilir.
9. Veritabanındaki bigindex tablosundan ilgili sözlük kaydına ait token'ın XML kaydı çekilir. Kayıt varsa bir sonraki adıma geçilir. Kayıt yoksa 11. Adıma geçilir.
10. Bigindex tablosunda ki token'a ait olan XML kaydı bir DataTable nesnesine çekilir. Ardından web sayfasının PageCollection tablosundaki "id" si ile token'nın sözlük içerisindeki frekans değeri iki farklı kolon ve bir satır olarak DataTable nesnesine eklenir. Sonrasında bu DataTable nesnesi tekrar XML kaydına çevrilerek, veritabanındaki bigindex tablosunun ilgili token'a ait XML alanı güncellenir ve 7. adıma gidilir.
11. docId ve frekans değeri kolonu olan bir DataTable nesnesi oluşturulur, ardından web sayfasının PageCollection tablosundaki "id" si ile token'nın sözlük içerisindeki frekans değeri iki farklı kolon ve bir satır olarak DataTable nesnesine eklenir. Sonrasında DataTable nesnesi tekrar XML kaydına çevrilerek, veritabanındaki bigindex tablosuna, sözlükten çekilen token ile birlikte eklenir ve 7. adıma gidilir.

12. Form içerisinde durdurma işleminin gerçekleşip gerçekleşmediğinin kontrolü yapılır. Durdurma işlemi gerçekleştirilimişse algoritma akışından çıkış gerçekleştirilir. Herhangi bir durdurma işlemi gerçekleştirilmemişse 3. adıma gidilir.

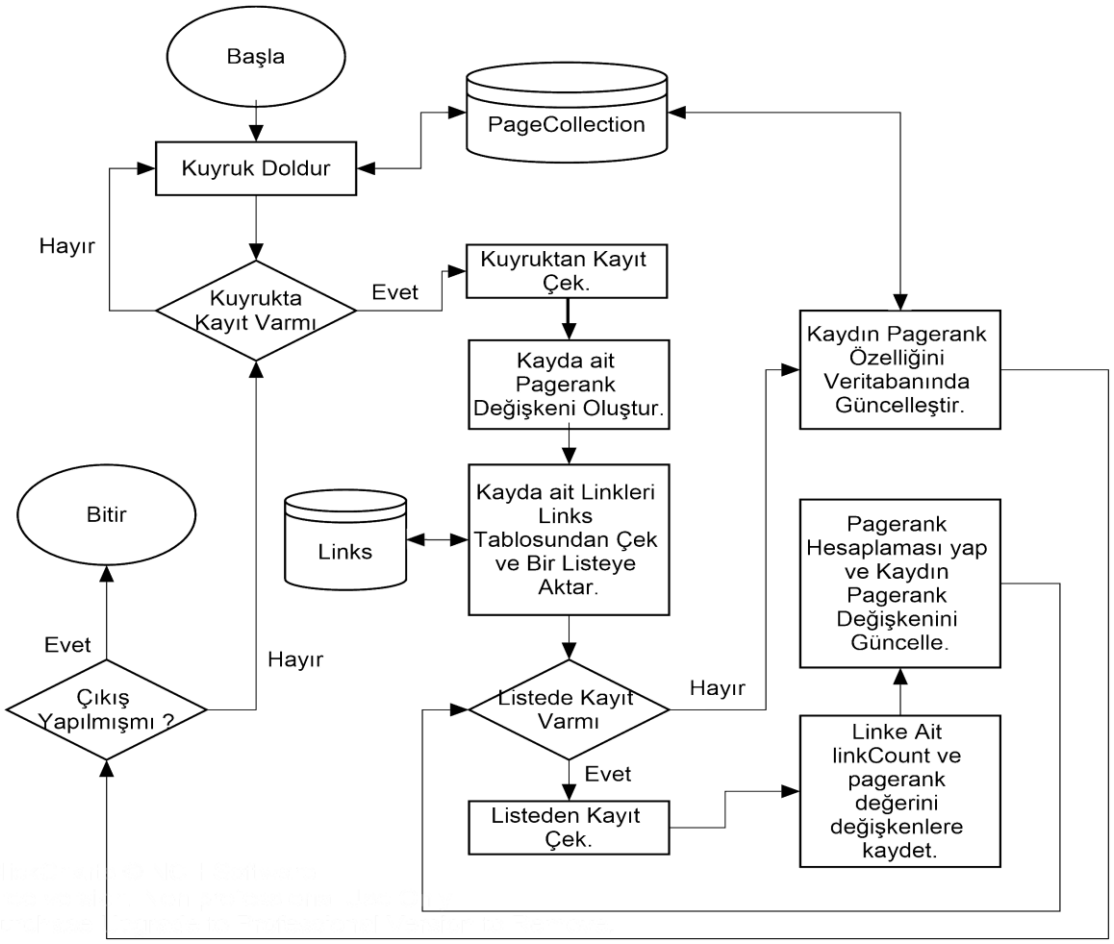
Yukarıdaki sıralamada WebIndexer uygulamasının akış diyagramı detaylı olarak adım adım anlatılmıştır. Daha detaylı bilgi için Ek B'de bulunan WebIndexer uygulamasının kodları incelenebilir. Ayrıca WebIndexer uygulamasında kullanılan ters indeks yapısı Şekil 4.7'de grafiksel olarak gösterilmektedir.




**Şekil 4.7** WebIndexer uygulamasının ters indeks yapısı

### 4.3 PageRankCalculator Uygulaması

PageRankCalculator uygulaması herhangi bir kullanıcı arayüzüne sahip değildir. Console Application olarak geliştirilmiştir. Görevi PageCollection tablosundaki bütün web sayfalarının pagerank değerlerini hesaplamaktır (Şekil 4.8).




**Şekil 4.8** PagerankCalculator Uygulamasının Akış Diyagramı

Kodları sonsuz bir döngü içerisinde olup, kullanıcı programdan çıkış yapıncaya kadar sürekli olarak web sayfalarının pagerank değerlerini güncellemektedir. Programın akış diyagramı Şekil 4.8'de görülmektedir. Diyagramda görüldüğü üzere PagerankCalculator uygulaması sırasıyla aşağıdaki adımları izleyerek çalışmaktadır.

1. İlk olarak veritabanındaki PageCollections tablosuna bakılarak, counted değeri 0 olan 1000 adet kayıt çekilir ve bir kuyruk yapısına aktarılır. Sonuç olarak herhangir satır döndürülmezse bütün PageCollections tablosu "UPDATE

pagecollections SET counted = 0;" sorgusu ile veritabanı güncellenir ve tekrar sorgu işlemi gerçekleştirilir.

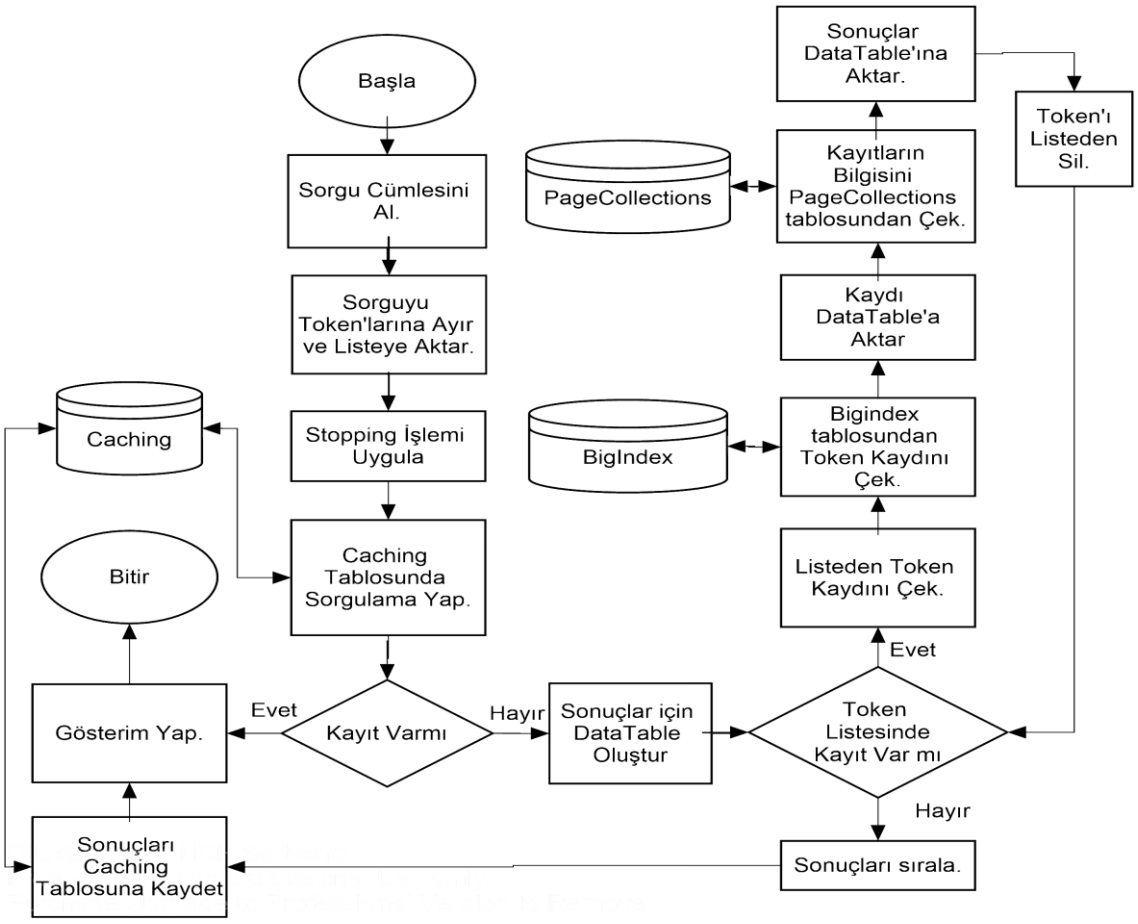
2. Kuyrukta kayıt var mı yok mu kontrol edilir ve kayıt varsa bir sonraki işleme geçilir. Kayıt yoksa 1. adıma gidilir.
3. Kuyruktan kayıt çekme işlemi gerçekleştirilir.
4. Kayıda ait bir pagerank değişkeni oluşturulur.
5. Kayda ait linkler veritabanındaki Links tablosundan çekilir ve string listesine aktarılır.
6. Listede kayıt var mı yok mu kontrol edilir. Kayıt varsa bir sonraki adıma geçilir. Kayıt yoksa 11. adıma gidilir.
7. Listeden kayıt çekme işlemi gerçekleştirilir.
8. Linke ait sayfa bilgileri PageCollections tablosundan çekilir.
9. Linke ait sayfa bilgilerinde bulunan linkCount ve pagerank değerleri hesaplamaya dahil edilebilmesi için birer değişkene aktarılır.
10. Linkin göstermiş olduğu kayıt üzerindeki pagerank etkisi hesaplanır ve kaydın pagerank değişkenine ekleme yapılır.
11. Kaydın veritabanındaki PageCollection tablosundaki kayıtlı olduğu satır bulunur ve bu satırın pagerank alanı, kayıt için oluşturulmuş ve güncellenmiş olan pagerank değişkeni ile güncelleştirilir. Ayrıca counted değeride 1 olarak güncelleştirilir.
12. Programdan çıkış yapılmış mı, yapılmamış mı kontrol edilir. Çıkış yapılmışsa program sonlandırılır. Çıkış yapılmamışsa 2. adıma gidilir.

Yukarıdaki sıralamada PagerankCalculator uygulamasının akış diyagramı detaylı olarak adım adım anlatılmıştır. Daha detaylı bilgi için Ek C'de bulunan PagerankCalculator uygulamasının kodları incelenebilir.

### 4.4 WebSearcher Uygulaması

WebSearcher uygulaması arama motorunun görünen yüzüdür. Kullanıcılar geliştirilen bu arama motorunu WebSearcher uygulaması aracılığıyla kullanabilirler. Bu bölümde arama motorunun kullanıcı arayüzü kısmından söz edilecektir. WebSearcher uygulaması adından da anlaşılacağı üzere, WebIndexer uygulaması tarafından indekslenmiş ve PagerankCalculator uygulaması tarafından pagerank değerleri

hesaplanmış olan web sayfalarını, kullanıcıların yapmış oldukları sorgulara göre listeleyen ve bazı kriterlere göre tekrar puanlayan ve sıralanmış sonuçları kullanıcılara gösteren bir web uygulamasıdır. Programın akış diyagramı Şekil 4.9'da görülmektedir.




**Şekil 4.9** WebSearcher uygulamasının akış diyagramı

Diyagramda görüldüğü üzere WebSearcher uygulaması sırasıyla aşağıdaki adımları izleyerek çalışmaktadır.

1. İlk olarak kullanıcıda sorgu cümlesi alınır.
2. Alınan sorgu cümlesi tokenlarına ayrılır ve bir listeye aktarılır.
3. Sorgu cümlesi kullanılarak veritabanındaki Caching tablosunda kayıt var mı yok mu kontrol edilir. Kayıt yoksa bir sonraki adıma geçilir. Kayıt varsa 13. adıma gidilerek sonuçların kullanıcıya gösterimi yapılır.
4. Sonuçlar için bir DataTable nesnesi oluşturulur.
5. Token listesinde kayıt var mı kontrol edilir. Kayıt varsa bir sonraki adıma gidilir. Kayıt yoksa 11. adıma gidilir.

6. Listeden token kaydı çekilir.
7. BigIndex tablosundan token'a ait kayıt çekilir.
8. Çekilen kayıttaki XML dosyası bir DataTable nesnesine aktarılır.
9. DataTable nesnesinde bulunan docId'ler kullanılarak PageCollections tablosundaki sayfalar çekilir ve sonuçlar DataTable'ına aktarılır.
10. Token kaydı listeden silinir ve 5. adıma gidilir.
11. Sonuçlar DataTable nesnesi içerisindeki sayfa listesi puanlarına göre sıralanır.
12. Sonuçlar DataTable nesnesi XML dosyasına çevrilir ve sorgu cümlesi ile birlikte Caching tablosuna kaydedilir.
13. Ardından elde edilen sonuçlar kullanıcıya gösterilir.

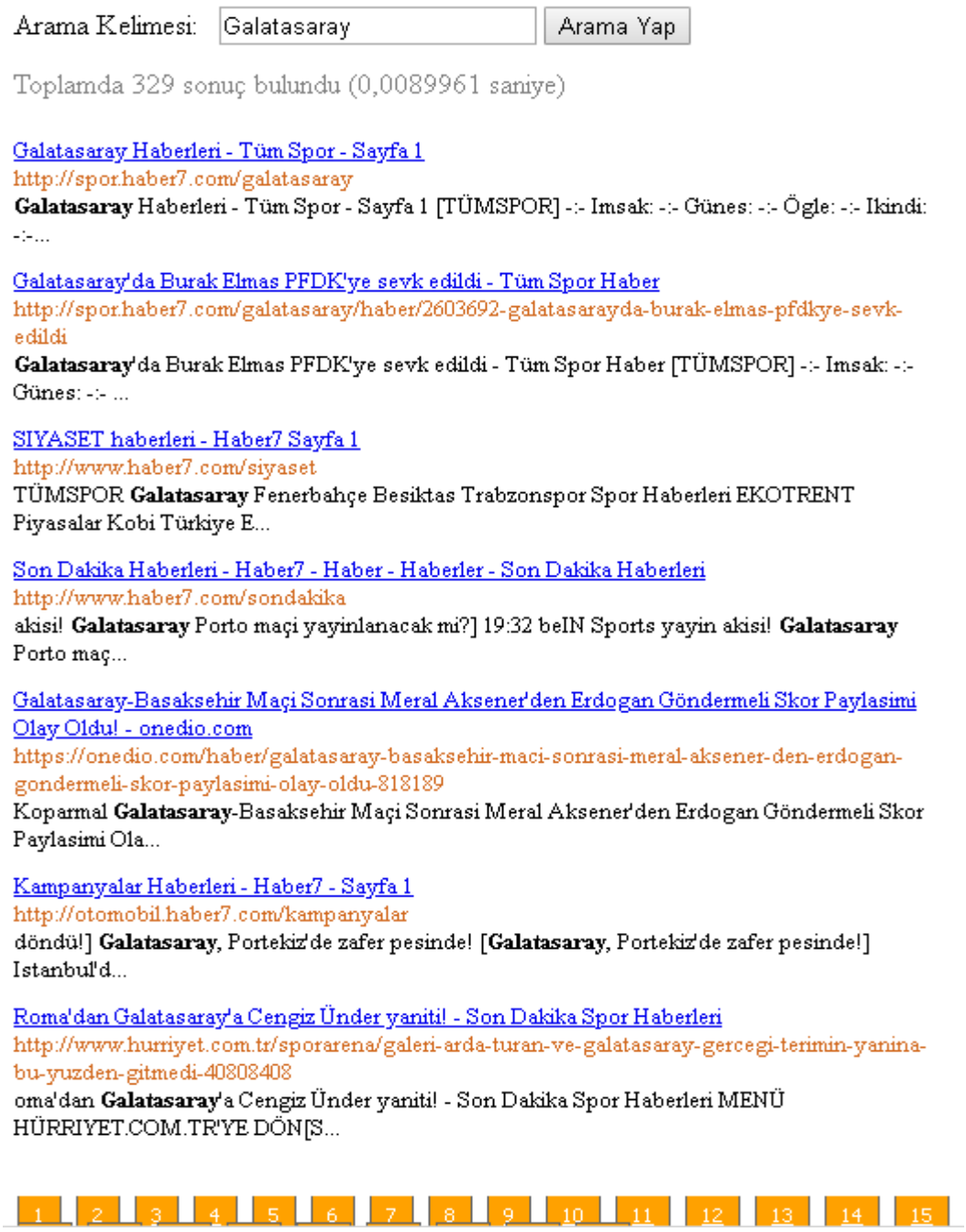
Yukarıdaki sıralamada WebSearcher uygulamasının akış diyagramı detaylı olarak adım adım anlatılmıştır. Arama motorları genelde web sayfalarının anlamlı bir şekilde sıralanması için sayfaların title ve url kısımlarında bulunan tokenları, web sayfalarının puanlanmasında birer kriter olarak kullanmaktadırlar (Croft et al., 2010). Web Searcher uygulamasında sonuçların anlamlı şekilde sıralanması için kullanılan formül denklem 4.1'de gösterilmektedir.

$$Rank = (1000xPR) + F + (100xT) + (100xU) \quad (4.1)$$

Denklemdeki terimlerin tanımları aşağıda verilmiştir.

- **Rank:** Sayfaların sıralanmasında kullanılacak sonuç değeri
- **PR:** Sayfanın pagerank değeri
- **F:** Sayfanın içerisinde aranılan tokenlara ait frekans değerleri toplamı
- **T:** Title içerisinde aranılan tokenların sayısı
- **U:** Url içerisinde aranılan tokenların sayısı

Daha detaylı bilgi için Ek D'de bulunan WebSearcher uygulamasının kodları incelenebilir. WebSearcher uygulamasının örnek sonuç sayfası Şekil 4.10'da gösterilmektedir.




**Şekil 4.10** WebSearcher uygulamasının örnek sonuç sayfası

## 4.4 Geliştirilen Arama Motorunun Sonuçlarının Karşılaştırılması

### 4.4.1 Apache Nutch ve Solr Karşılaştırması

Geliştirilen arama motoruyle kıyaslama yapabilmek amacıyla, Apache Nutch ve Solr kütüphaneleri biraraya getirilerek bir arama motoru oluşturulmuş ve farklı sorgular

kullanılarak kıyaslamalar yapılmıştır. Kıyaslamalar yapılırken, geliştirilen arama motoru içerisinde, Caching yapısı kullanılarak sorgulamalar gerçekleştirilmiştir.

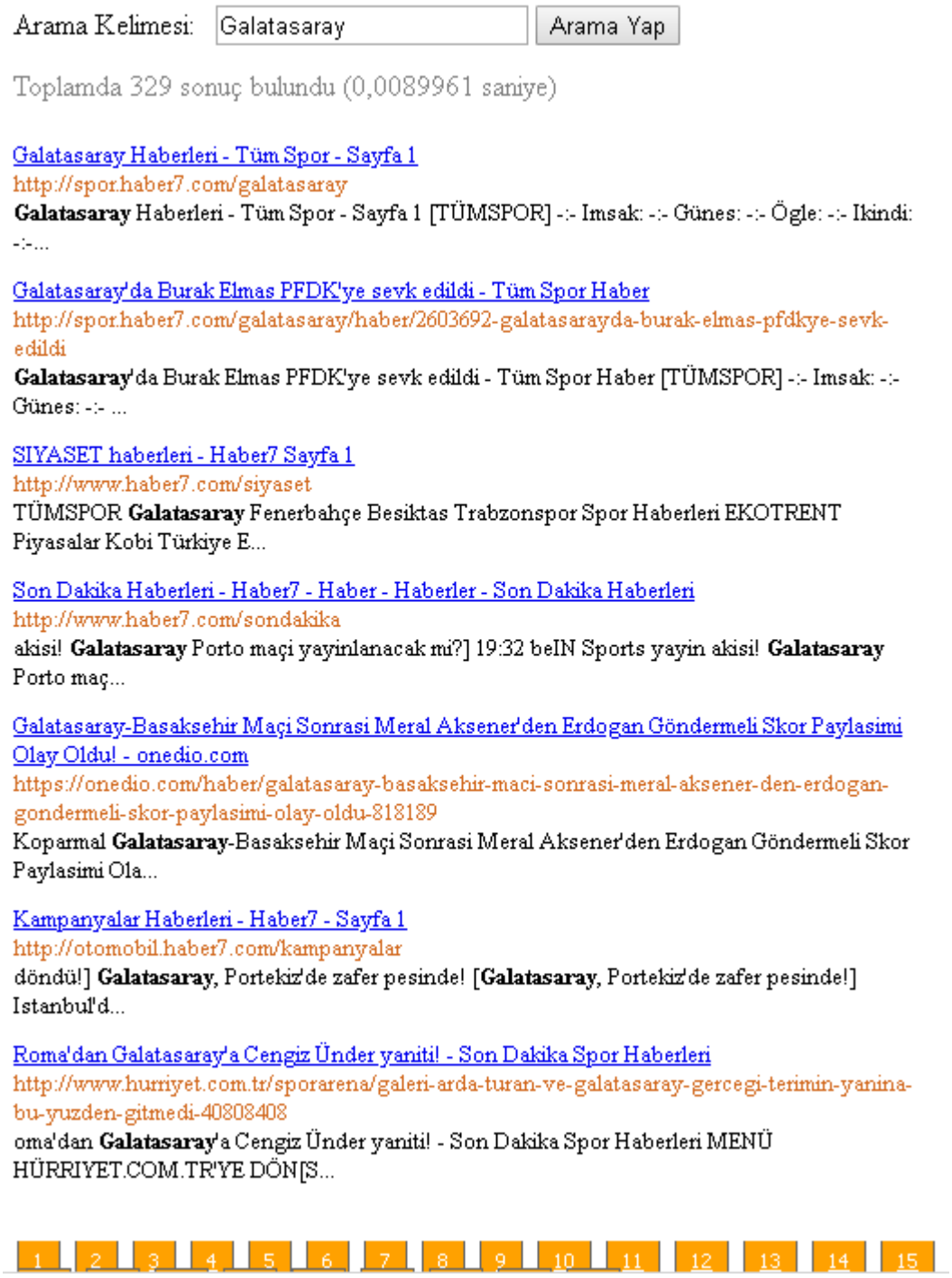
"Ahmet" sorgusu her iki arama motorunda da sorgulanmıştır. Şekil 4.11'da Apache Nutch ve Solr kütüphaneleri ile geliştirilen arama motoru sonuçları gösterilmektedir. Şekil 4.12'de tez kapsamında geliştirilen ve Caching yapısı kullanılan arama motoru sonuçları gösterilmektedir.


**Şekil 4.11** "Ahmet" sorgusunun Apache Nutch ve Solr kullanılan arama motorundaki sonucu


**Şekil 4.12** "Ahmet" sorgusunun tez kapsamında geliştirilen arama motorundaki sonucu

"Galatasaray" sorgusu her iki arama motorunda da sorgulanmıştır. Şekil 4.13'de Apache Nutch ve Solr kütüphaneleri ile geliştirilen arama motoru sonuçları gösterilmektedir. Şekil 4.14'te tez kapsamında geliştirilen ve Caching yapısı kullanılan arama motoru sonuçları gösterilmektedir.

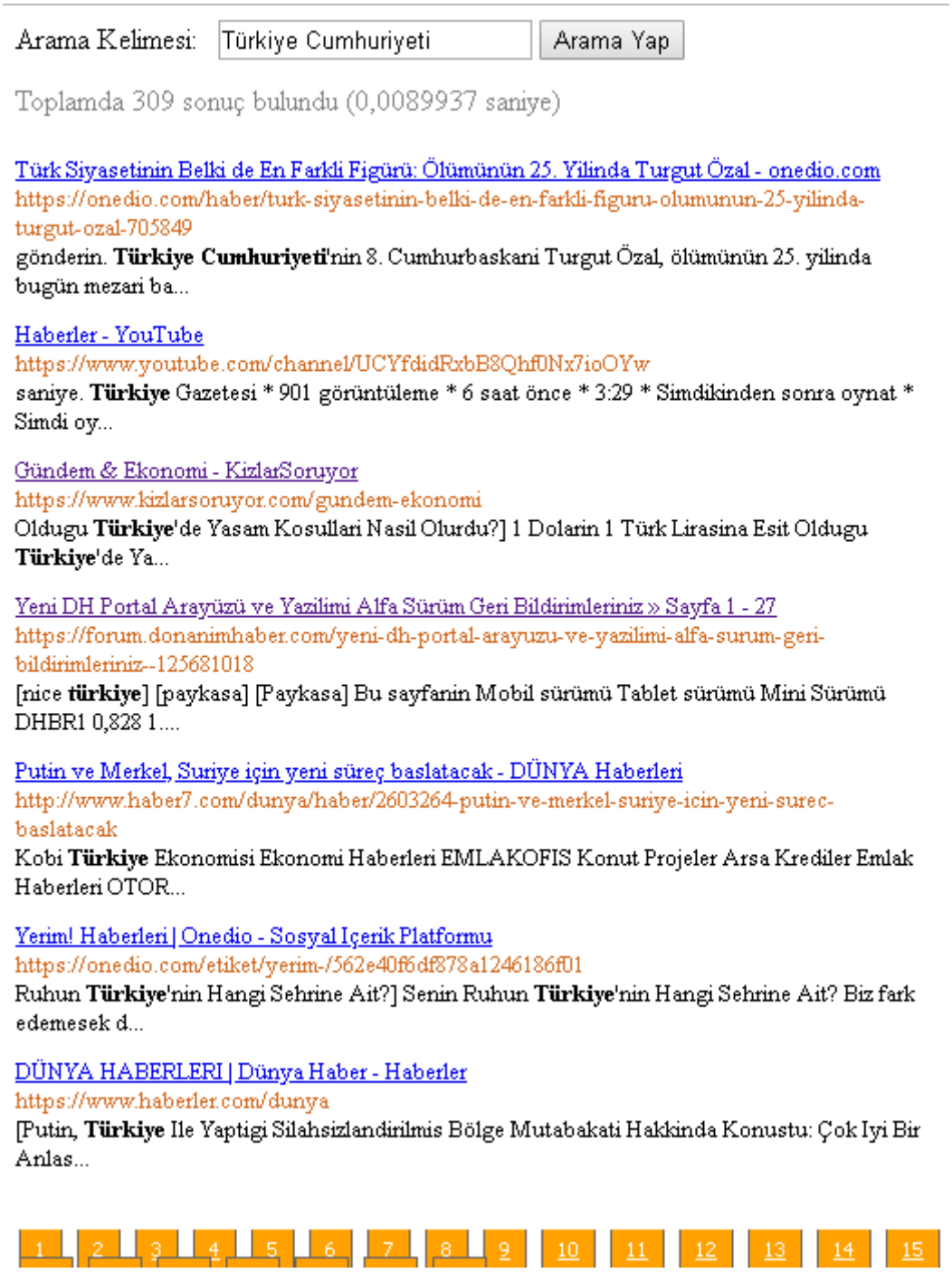

**Şekil 4.13** "Galatasaray" sorgusunun Apache Nutch ve Solr kullanılan arama motorundaki sonucu




**Şekil 4.14** "Galatasaray" sorgusunun tez kapsamında geliştirilen arama motorundaki sonucu

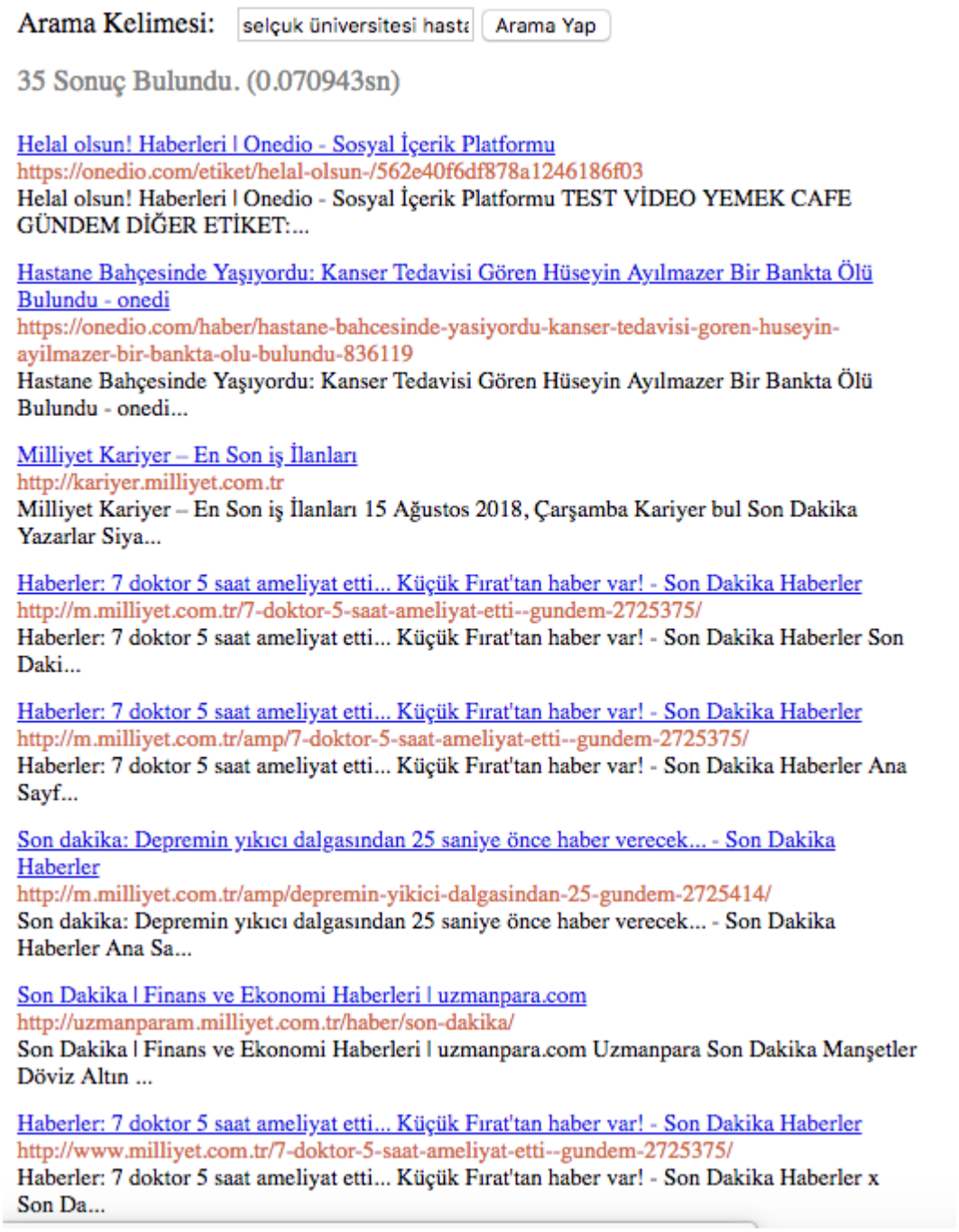
"Türkiye Cumhuriyeti" sorgusu her iki arama motorunda da sorgulanmıştır. Şekil 4.15'te Apache Nutch ve Solr kütüphaneleri ile geliştirilen arama motoru sonuçları gösterilmektedir. Şekil 4.16'da tez kapsamında geliştirilen ve Caching yapısı kullanılan arama motoru sonuçları gösterilmektedir.


**Şekil 4.15** "Türkiye Cumhuriyeti" sorgusunun Apache Nutch ve Solr kullanılan arama motorundaki sonucu

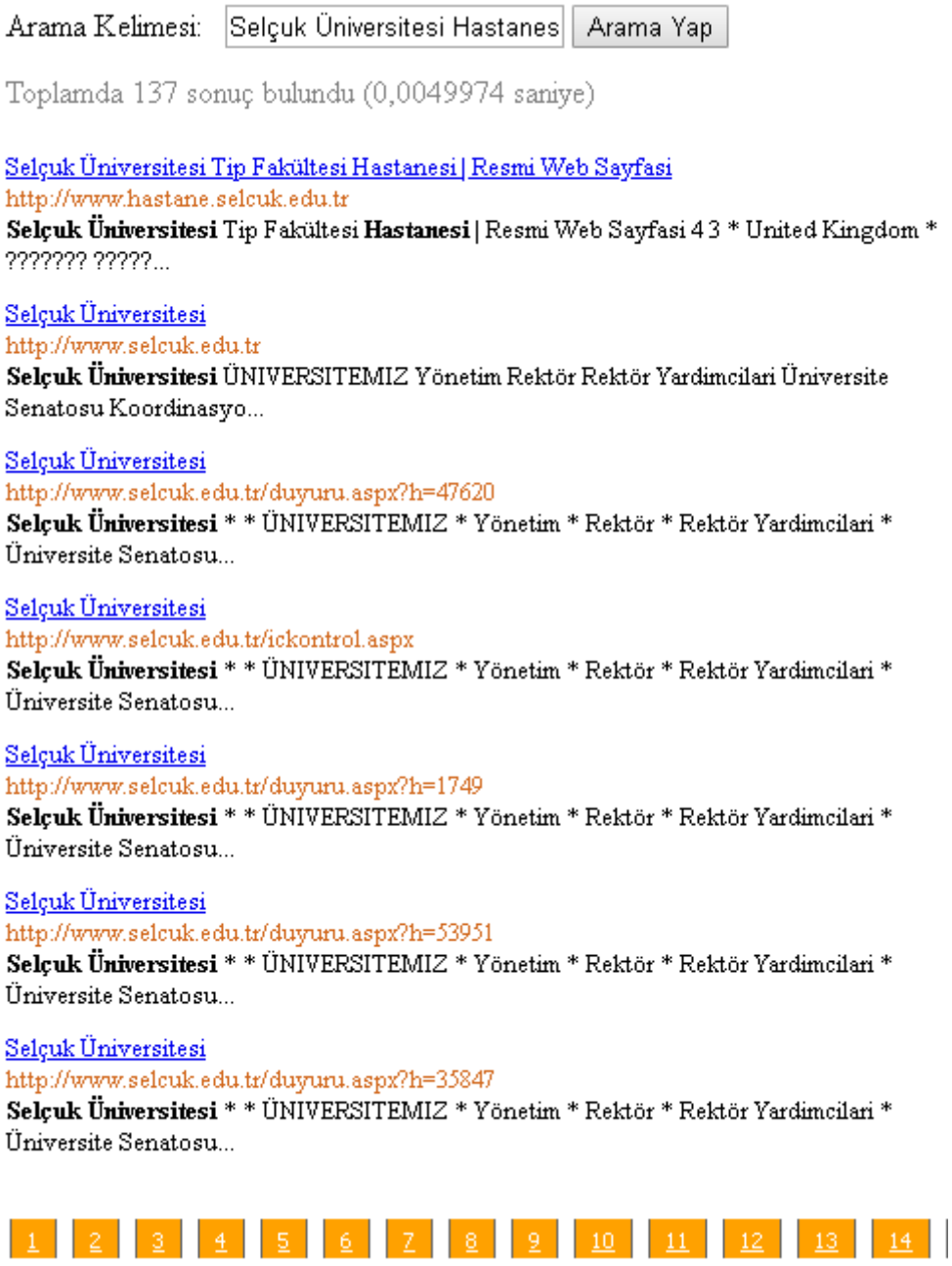



**Şekil 4.16** "Türkiye Cumhuriyeti" sorgusunun tez kapsamında geliştirilen arama motorundaki sonucu

"Selçuk Üniversitesi Hastanesi" sorgusu her iki arama motorunda da sorgulanmıştır. Şekil 4.17'de Apache Nutch ve Solr kütüphaneleri ile geliştirilen arama motoru sonuçları gösterilmektedir. Şekil 4.18'te tez kapsamında geliştirilen ve Caching yapısı kullanılan arama motoru sonuçları gösterilmektedir.



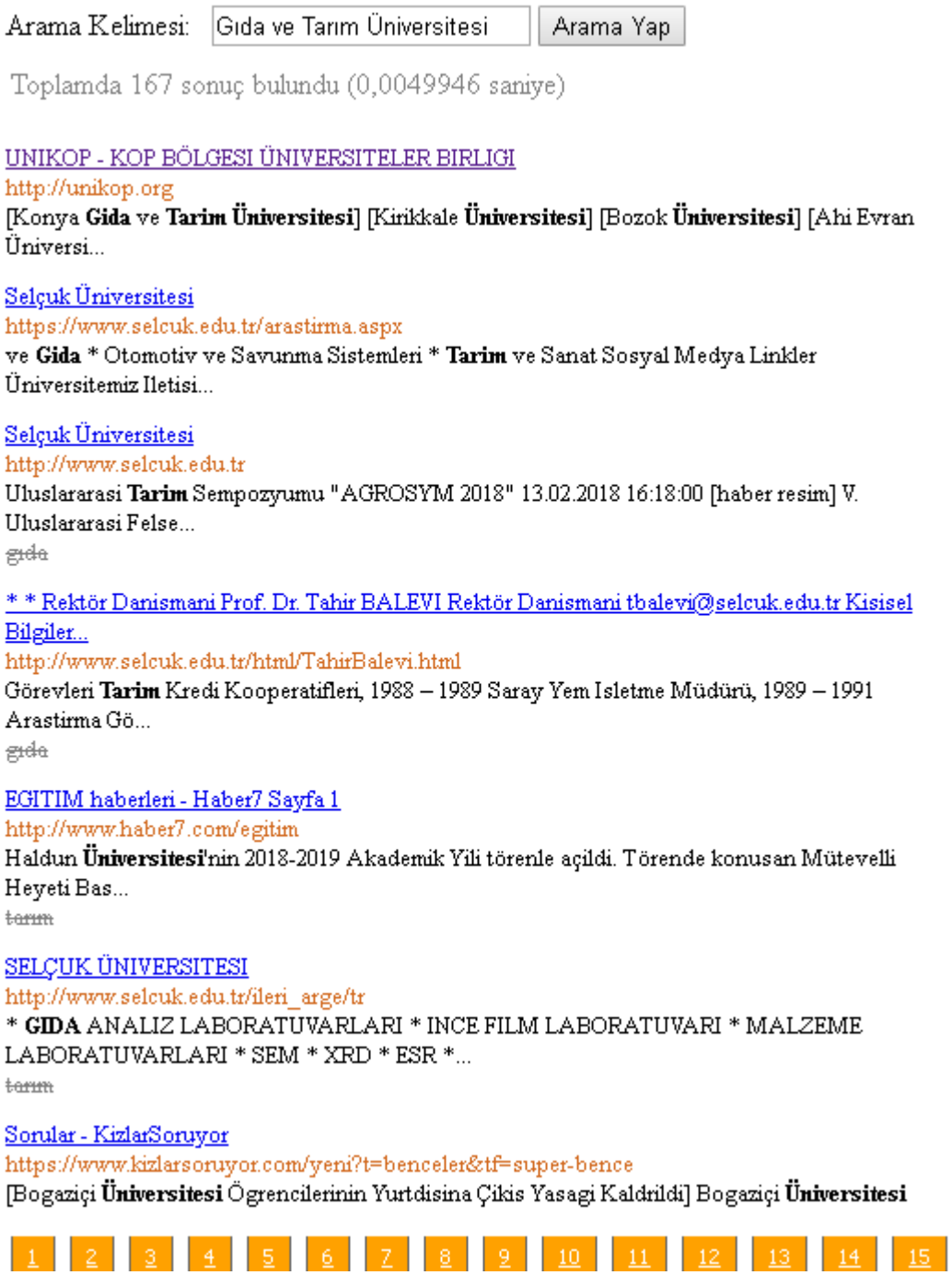

**Şekil 4.17** "Selçuk Üniversitesi Hastanesi" sorgusunun Apache Nutch ve Solr kullanılan arama motorundaki sonucu




**Şekil 4.18** "Selçuk Üniversitesi Hastanesi" sorgusunun tez kapsamında geliştirilen arama motorundaki sonucu

"Gıda ve Tarım Üniversitesi" sorgusu her iki arama motorunda da sorgulanmıştır. Şekil 4.19'da Apache Nutch ve Solr kütüphaneleri ile geliştirilen arama motoru sonuçları gösterilmektedir. Şekil 4.20'de tez kapsamında geliştirilen ve Caching yapısı kullanılan arama motoru sonuçları gösterilmektedir.


**Şekil 4.19** "Gıda ve Tarım Üniversitesi" sorgusunun Apache Nutch ve Solr kullanılan arama motorundaki sonucu




**Şekil 4.20** "Gıda ve Tarım Üniversitesi" sorgusunun tez kapsamında geliştirilen arama motorundaki sonucu

### 4.4.2 Sonuçların Kıyaslamaları

Galatasaray sorgusunun sonuçları incelendiğinde Apache Nutch ve Solr kütüphaneleri kullanılarak geliştirilen arama motorunda en anlamlı sonucun 6. sırada yer alan http://m.milliyet.com.tr/amp/galatasaray-a-carole-mujdesi-750-galatasaray-2725372-skorerhaber/ sayfasının getirildiği görülmektedir. Listelenen diğer sonuçlar ele alındığında bu sırada gelen sonuçtan daha kaliteli sonuçların alt sayfalarda yer aldığı görülmektedir. Tez kapsamında geliştirilen arama motorunda ise en anlamlı sonucun 1. sırada yer alan http://spor.haber7.com/galatasaray sayfasının getirildiği görülmektedir. Sayfa incelendiğinde tamamen "Galatasaray" takımı için oluşturulmuş ve içerisinde birçok "Galatasaray" haberinin yer aldığı bir sayfa olduğu gözlemlenmiştir. Apache Nutc ve Solr arama motorunun listelediği sonuçlardan en anlamlı sonucun 6. Sırada yer almasının yanı sıra sadece "Galatasaray" hakkında bir haberin yer aldığı bir sayfa olduğu görülmektedir.

Türkiye Cumhuriyeti sorgusunun sonuçları incelendiğinde Apache Nutch ve Solr kütüphaneleri kullanılarak geliştirilen arama motorunun "Türkiye Cumhuriyeti" sorgusunda "Türkiye" ve "Cumhuriyeti" kelimelerinin sorgularını birleştirip gösterdiği gözlemlenmiş olup bu iki kelimenin yanyana bir bütün halinde gösterildiği bir sonuç bulunamamıştır. Tez kapsamında geliştirilen arama motorunda ise ilk başta sorgu cümlesinin tamamının yer aldığı sonuçlar ilk sırada gösterilmiştir.

Selçuk Üniversitesi Hastanesi ve Gıda ve Tarım Üniversitesi sorgularında, Apache Nutch ve Solr kütüphanesi kullanılarak geliştirilen arama motorunda, yapılan sorguların bir bütün halinde işlenmediği ve anlamlı sonuçlar üretmediği görülmektedir. Tez kapsamında geliştirilen arama motorunda ise "Selçuk Üniversitesi Hastanesi" sorgusunda bulunmak istenildiği üzere Selçuk Üniversitesi Hastanesinin web sayfası ilk sırada gösterilmiştir. Gıda ve Tarım Üniversitesi sorgusunda da gene bu üniversitenin isminin yer aldığı sayfanın ilk sırada listelendiği gözlemlenmiştir.

Arama sonuçları, iki arama motorunun süreleri açısından ve sıralama sonuçları açısından karşılaştırıldığında tez kapsamında geliştirilen arama motorunun daha hızlı sürelerde, daha anlamlı sonuçlar elde ettiği görülmektedir.

# 5. SONUÇLAR VE ÖNERİLER

## 5.1 Sonuçlar

Arama motorları internet üzerindeki büyük boyuttaki veri ile insanlar arasında bir köprü oluşturan, insanların istedikleri bilgilere ulaşmasını sağlayan sistemlerdir. Arama motorları, web üzerindeki bütün web sitelerinin bir kopyasını alır ve kopyasını aldığı bu web sitelerinin içerisindeki bilgiyi analiz ve organize eder. Ardından kullanıcılara anlamlı ve aradıklarını bulabilecekleri sonuçları üretmek için web sayfalarının kayıtlı olduğu veritabanlarından sayfaları çeker ve kullanıcıya listeleme işlemi yaparlar. Buradan da anlaşılacağı üzere, kullanıcıların arama motorunda arama yaptığında aradıkları bilgiye ait web sayfaları arama motorunun veritabanında kayıtlı olan sayfalardır. Arama motorları kendi kayıtlarından sorgu yaparak kullanıcılara listeleme yaparlar. Bu yüzden arama motorlarının gösterdikleri sonuçlar güncel sonuçlar olmayabilir, web sitesi değişmiş veya kaldırılmış olabilir. Arama motorlarının veritabanları sürekli değiştiğinden ve geliştirildiğinden dolayı veritabanlarında yapmış oldukları işlemler sürekli olarak devam eden ve sürekli güncellenen işlemlerdir.

Arama motorları, kaydedebilecekleri kadar en büyük veriyi kaydetmeye çalışırlar ama web üzerindeki bütün web sayfalarını çekip, indeksleyemezler. Çünkü burada sürekli güncellenen ve sürekli büyüyen çok büyük bir veriden bahsedilmektedir. Arama motorları, bu büyük veri içerisinde kullanıcıların yaptığı sorgularla alakalı web sayfalarını bulmalı ve bu elde edilmek istenen sonuçları hızlı ve anlamlı bir şekilde kullanıcılara göstermek için düzenleyebilmelidirler. Bu tez içerisinde arama motorlarının tarihi, neden gereksinim duyuldukları ve mimarisi detaylı şekilde anlatılmıştır. Arama Motoru Mimarisi açısından temel dört bileşen üzerinde durulup bu dört bileşen hakkında detaylı teknik bilgiler verilmiş ve bu dört bileşen uygulama olarak geliştirilmiştir.

Bahsedilen bu dört yapı geliştirilirken C#, Asp.Net ve MySQL teknolojileri kullanılmıştır. Türkiye'de arama motorlarının kullandığı bu dört yapının tamamının açık kaynak olarak kodlandığı ve paylaşıldığı bir tez veya proje tespit edilmediğinden dolayı Bilgisayar Mühendisliği okuyan veya bu alanda yüksek lisans yapan öğrencilerin arama motoru mimarisini anlamaları açısından faydalanabileceği veya kendi arama motorlarını oluşturabilecekleri bir proje geliştirilmiştir. Genelde, bu dört yapının yabancı kaynaklı şirketler tarafından açık kaynak olarak geliştirildiği ve bir kütüphane olarak kullanıma

sunulduğu görülmektedir. Bu sistemleri ticari kullanım için değilde daha çok deneysel çalışma olarak değerlendirmek daha doğrudur. Örneğin, geliştirilen bu sistemlerden biri olan Apache Nutch uygulaması stabil çalışan bir uygulama değildir, çünkü yapılan çalışmalarda, Apache Nutch uygulamasının birçok kez kullanıcı tarafından sonlandırma işlemi gerçekleşmeden, uygulamayı kendi kendine sonlandırdığı görülmüştür. Apache Solr uygulamasının ise hız açısından yüksek hızlı çalışmasına rağmen, sorgulamaların sonucunda anlamlı sıralamalar içeren sonuçlar listelemediği tespit edilmiştir.

## 5.2 Öneriler

Türkiye'de yapılan tez çalışmalarında genelde hazır kütüphanelerin kullanıldığı görülmüştür. Bu açıdan bakıldığında bu arama motoru, açık kaynak olarak geliştirilmiş ve bu yönde kendini geliştirmek isteyen araştırmacılara da bir kaynak olarak github sitesinde, "https://github.com/mehmetkarlik/search_engine" adresi altında, açık kaynak kodlu olarak paylaşılmıştır.

Geliştirilen bu arama motoru, temel düzeyde bir web arama motorudur. Google, Yandex veya Bing gibi büyük web arama motorlarıyla kıyaslandığında birçok eksik özelliği bulunmaktadır. Bu eksik özelliklerin zaman içerisinde tamamlanması planlanmaktadır. Gelecekte yapılması planlanan özellikler sırasıyla şu şekilde listelenmiştir.

1. Bölgesel Arama
2. Kelime Tahmin Sistemi
3. Dağıtık Mimari
4. Görsel Arama
5. Harita Desteği

**KAYNAKLAR**

Anonim. (2012, 05.10.2012). Pagerank Nedir? Retrieved 05.05.2018, 2018, from https://internetreklamevi.com/page-rank-nedir/

Anonim. (2017a). Arama Motoru. Retrieved 4 Ekim, 2017, from http://www.turkcewiki.org/wiki/Arama_motoru

Anonim. (2017b, 05.06.2017). The Basics: Working with Nutch 2x. Retrieved 05.04.2018, 2018, from https://www.mobomo.com/2017/06/the-basics-working-with-nutch-2-x/

Anonim. (2017c). Geliyoo. Retrieved 4 Aralık, 2017, from http://www.turkcewiki.org/wiki/Geliyoo

Anonim. (2017d). Google. Retrieved 15 Ekim, 2017, from http://www.turkcewiki.org/wiki/Google

Anonim. (2017e). Yaani. Retrieved 18 Kasım, 2017, from https://www.turkcell.com.tr/servisler/yaani

Anonim. (2017f). Yandex. Retrieved 18 Kasım, 2017, from http://www.turkcewiki.org/wiki/Yandex

Anonim. (2018a). Deep web What is it and hot to access it. Retrieved 05.08.2018, 2018, from https://darkwebnews.com/deep-web/

Anonim. (2018b, 05.05.2018). Internet Connectivity. Retrieved 04.08.2018, 2018, from https://www.conceptdraw.com/How-To-Guide/internet-connectivity

Anonim. (2018c, 01.01.2018). Nutch. Retrieved 18.09.2018, 2018, from http://www.turkcewiki.org/wiki/Nutch

Ashley, J. (2018, 2018). How many web pages on the internet? Retrieved 08.09.2018, 2018, from https://askwonder.com/q/how-many-web-pages-are-on-the-internet-presently-57336062ded4a34e0083c6b0

Bacak, A. (2016). Veri İndeksleme Yöntemleri. Retrieved 30 Ekim, 2017, from https://blog.crypttech.com/2016/12/veri-indeksleme-yontemleri.html

Baeza-Yates, R., Gionis, A., Junqueira, F. P., Murdock, V., Plachouras, V., & Silvestri, F. (2008). Design trade-offs for search engine caching. *ACM Transactions on the Web (TWEB), 2*(4), 20.

Baker, P. R., Trinidad, J. C., & Chalkley, R. J. (2011). Modification site localization scoring integrated into a search engine. *Molecular & Cellular Proteomics, 10*(7), M111. 008078.

Barroso, L. A., Dean, J., & Holzle, U. (2003). Web search for a planet: The Google cluster architecture. *IEEE micro, 23*(2), 22-28.

Benczur, A. A., Csalogany, K., Sarlos, T., & Uher, M. (2005). *Spamrank–fully automatic link spam detection work in progress.* Paper presented at the Proceedings of the first international workshop on adversarial information retrieval on the web.

Bergman, M. K. (2001). White paper: the deep web: surfacing hidden value. *Journal of electronic publishing, 7*(1).

Bernstein, Y., & Zobel, J. (2005). *Redundant documents and search effectiveness.* Paper presented at the Proceedings of the 14th ACM international conference on Information and knowledge management.

Blanchard, A. (2007). Understanding and customizing stopword lists for enhanced patent mapping. *World Patent Information*, *29*(4), 308-316.

Borthakur, D. (2007). The hadoop distributed file system: Architecture and design. *Hadoop Project Website, 11*(2007), 21.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems, 30*(1-7), 107-117.

Bulut, B. (2014, 04.03.2014). Apache Solr Nedir? Retrieved 11.9.2018, 2018, from https://betulblt.wordpress.com/2014/03/04/apache-solr-nedir/

Büttcher, S., Clarke, C. L., & Cormack, G. V. (2016). *Information retrieval: Implementing and evaluating search engines*: Mit Press.

Castillo, C. (2005). *Effective web crawling.* Paper presented at the Acm sigir forum.

Catlett, C. E. (1989). The NFSNET: Beginnings of a National Research Internet. *Academic Computing, 3*(5).

Chakrabarti, S., Van den Berg, M., & Dom, B. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *Computer networks, 31*(11-16), 1623-1640.

Chen, A., & Gey, F. C. (2002). *Building an Arabic Stemmer for Information Retrieval.* Paper presented at the TREC.

Cho, J., & Garcia-Molina, H. (2002). *Parallel crawlers.* Paper presented at the Proceedings of the 11th international conference on World Wide Web.

Cho, J., & Garcia-Molina, H. (2003). Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems (TODS)*, *28*(4), 390-426.

Croft, W. B., Metzler, D., & Strohman, T. (2010). *Search engines: Information retrieval in practice* (Vol. 283): Addison-Wesley Reading.

DEEPWEBADMIN. (2016). How Big is the Deep Web. Retrieved Mayıs, 2018, from (https://www.deepweb-sites.com/how-big-is-the-deep-web/)

Emyan, A. (2014). Baidu Nedir, Ne İşe Yarar. Retrieved 4 Kasım, 2017, from https://teknovi.com/baidu-nedir-ne-ise-yarar

Fujii, A., & Ishikawa, T. (1999). Cross-language information retrieval for technical documents. *arXiv preprint cs/9907007*.

Greene, R. (2000). Web work: A history of Internet art. *Artforum, 38*(9), 162-167.

Gunes, C. (2014, 12.07.2014). İnternet Tarayıcısı Nedir? Retrieved 05.04.2018, 2018, from https://cangunes.com/internet-tarayicisi-nedir/)

Hearst, M. A. (1999). *Untangling text data mining.* Paper presented at the Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics.

Huberman, B. A., & Adamic, L. A. (1999). Internet: growth dynamics of the world-wide web. *Nature, 401*(6749), 131.

Jain, A. (2013). The Role and Importance of Search Engine and Search Engine Optimization. *International Journal of Emerging Trends & Technology in ComputerScience, 2*(3), 99-102.

Kausar, M. A., Dhaka, V., & Singh, S. K. (2013). Web crawler: a review. *International Journal of Computer Applications, 63*(2).

Kemp, S. (2017, 11.04.2017). The global state of the internet. Retrieved 04.03.2018, 2018, from https://thenextweb.com/contributors/2017/04/11/current-global-state-internet/

Kemp, S. (2018, 30.01.2018). Digital in 2018: World's internet users pass the 4 billion mark - We Are Social. Retrieved 10.09.2018, 2018, from https://wearesocial.com/blog/2018/01/global-digital-report-2018

Kirsch, S. T., Chang, W. I., & Miller, E. R. (1999). Real-time document collection search engine with phrase indexing: Google Patents.

Larson, R. R. (2010). *Introduction to information retrieval* (Vol. 61).

Liu, T.-Y. (2009). *Learning to rank for information retrieval* (Vol. 3).

Maeda, A., Sadat, F., Yoshikawa, M., & Uemura, S. (2000). *Query term disambiguation for Web cross-language information retrieval using a search engine.* Paper presented at the Proceedings of the fifth international workshop on on Information retrieval with Asian languages.

Mahapatra, A. K., & Biswas, S. (2011). Inverted indexes: Types and techniques. *Int. Journal of Computer Science, 8*.

Manios, C. (2015, 16.10.2015). Introduction to Apache Solr. Retrieved 18.09.2018, 2018, from https://www.slideshare.net/ChristosManios/introduction-to-apache-solr-54076189

McQuillan, J., Richer, I., & Rosen, E. (1980). The new routing algorithm for the ARPANET. *IEEE transactions on communications, 28*(5), 711-719.

Olston, C., & Najork, M. (2010). Web crawling. *Foundations and Trends in Information Retrieval, 4*(3), 175-246.

Reddy, D. (2017, 01.09.2017). Lucene Architecture. Retrieved 09.07.2018, 2018, from http://placetoshareall.blogspot.com/2017/09/lucene-architecture-image.html

Roshdi, A., & Roohparvar, A. (2015). Information Retrieval Techniques and Applications. *International Journal of Computer Networks and Communications Security, 3*(9).

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management, 24*(5), 513-523.

Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39): Cambridge University Press.

Segal, B. (1995). A short history of Internet protocols at CERN. *Professional webpage April http://ben.home.cern.ch/ben/TCPHIST.html*.

Tong, S., Lerner, U., Singhal, A., Haahr, P., & Baker, S. (2008). Locating meaningful stopwords or stop-phrases in keyword-based retrieval systems: Google Patents.

Van Rijsbergen, C. (1979). *Information retrieval* (Vol. 14): . dept. of computer science, university of glasgow.

Wilson, J. (2008). Method for enabling dynamic websites to be indexed within search engines: Google Patents.

Yadav, D., Sharma, A., Gupta, J., Garg, N., & Mahajan, A. (2007). *Architecture for parallel crawling and algorithm for change detection in web pages.* Paper presented at the Information Technology,(ICIT 2007). 10th International Conference on.

Yeşilkaya, C. (2013, 03.03.2013). Apache Solr | Kurulumu | Örnek Sorgulama. Retrieved 08.10.2018, 2018, from https://kodcu.com/2013/03/apache-solr-kurulumu-ornek-sorgulama/

Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS), 22*(2), 179-214.

Zobel, J., & Moffat, A. (2006). Inverted files for text search engines. *ACM computing surveys (CSUR), 38*(2), 6.

# EKLER

## EK A - WEBCRAWLER UYGULAMASI

### Form1.cs

```
using HtmlAgilityPack;
using MongoDB.Bson;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Security.Permissions;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WebCrawler
{
   public partial class Form1 : Form
   {
      bool kuyrukBosmu;
      int taskCount = 0;

      List<TaskClass> taskListesi = new List<TaskClass>();
      ManualResetEvent mrse = new ManualResetEvent(false);
      ConcurrentQueue<string> urlList = new ConcurrentQueue<string>();

      public Form1()
      {
         InitializeComponent();

      }

      private void Form1_Load(object sender, EventArgs e)
      {
         DataTable dt = Database.get1000RecordInLinkCollection();
         for (int i = 0; i < dt.Rows.Count; i++)
         {
            urlList.Enqueue(dt.Rows[i]["url"].ToString());
         }
```

```
            kuyrukBosmu = false;

            BasaDon();
        }

        public void BasaDon()
        {

            this.Invoke(new MethodInvoker(delegate ()
            {
                btnDuraklat.Enabled = false;
                btnDevam.Enabled = false;
                btnDurdur.Enabled = false;
                btnBaslat.Enabled = true;

            }));

            //Database.deleteDatasInCollections();
            //for (int i = 0; i < lstUrlList.Items.Count; i++)
            //{
            //    Database.addLink(lstUrlList.Items[i].ToString(), null);
            //}

            this.Invoke(new MethodInvoker(delegate ()
            {
                lstUrlList.HorizontalScrollbar = true;
                lstDurum.HorizontalScrollbar = true;
                //timer1.Interval = 10;

                //timer1.Start();
                lstDurum.Items.Clear();
            }));
        }

        public void Resume()
        {
            mrse.Set();
        }

        public void Pause()
        {
            mrse.Reset();
        }

        [SecurityPermissionAttribute(SecurityAction.Demand, ControlThread =
true)]
        private void KillTheTasks()
        {
            Pause();
```

```
    for (int i = 0; i < taskListesi.Count; i++)
    {
        TaskClass t = taskListesi[i];
        t.cancelTokenSource.Cancel();
    }

    taskListesi = new List<TaskClass>();
}

private void btnBaslat_Click(object sender, EventArgs e)
{

    if (txtTaskCount.Text != "")
        taskCount = int.Parse(txtTaskCount.Text);

    if (taskCount > 100 || taskCount < 1)
    {
        MessageBox.Show("1 ile 100 aralığında bir değer giriniz.");
        txtTaskCount.Text = "";
        return;
    }


    CancellationTokenSource cts = new CancellationTokenSource();
    CancellationToken ct = cts.Token;
    Task task = new Task(taskStart, ct);
    //task.IsBackground = true;

    task.Start();

    taskListesi.Add(new TaskClass(task, cts));
    txtTaskCount.Enabled = false;
    btnBaslat.Enabled = false;
    btnDuraklat.Enabled = true;
    btnDurdur.Enabled = true;
    Resume();
}

private void taskStart()
{
    taskListesi = new List<TaskClass>();
    for (int i = 0; i < taskCount; i++)
    {
        mrse.WaitOne();
        CancellationTokenSource cts = new CancellationTokenSource();
        CancellationToken ct = cts.Token;
        Task task = new Task(crawl, ct);
        //th.IsBackground = true;
        task.Start();
        taskListesi.Add(new TaskClass(task, cts));
```

```
        }
    }

    private void crawl()
    {

        while (true)
        {
            mrse.WaitOne();
            string url = "";
            try
            {
                Random rastgele = new Random();

                Thread.Sleep(rastgele.Next(1, 10) * 1000 + rastgele.Next(1, 10) *
100 + rastgele.Next(1, 10) * 10 + rastgele.Next(1, 10));
                if (kuyrukBosmu == true)
                    continue;

                if (urlList.Count == 0)
                {
                    kuyrukBosmu = true;
                    Pause();
                    DataTable dt = Database.get1000RecordInLinkCollection();
                    for (int i = 0; i < dt.Rows.Count; i++)
                    {
                        urlList.Enqueue(dt.Rows[i]["url"].ToString());
                    }
                    kuyrukBosmu = false;
                    Resume();
                }
                urlList.TryDequeue(out url);

                if (url == "" || url.Contains("Başarısız Kayıt"))
                {
                    if (url != "")
                    {
                        throw new Exception(url);
                    }
                    continue;
                }


                try
                {

                    HttpWebRequest webRequest = null;
                    webRequest = HttpWebRequest.Create(url) as HttpWebRequest;
                    webRequest.Accept = "text/html";
```

```
                webRequest.Timeout = 60000;
                webRequest.Method = "GET";

                WebResponse webResponse = null;
                webResponse = webRequest.GetResponse();

                string url2 = webResponse.ResponseUri.ToString();

                if (url != url2)
                {
                    using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
                    {
                        Database.deleteLinkInLinkCollections(url, connection);
                        Database.deleteLinkInLinks(url, connection);
                        Database.deleteLinkInPageCollections(url, connection);
                    }
                    url = url2;
                }

                if (url[url.Length - 1] == '/')
                {
                    url = url.Remove(url.Length - 1);
                }
                var Date = webResponse.Headers["Last-Modified"];

                if (Date == null)
                {
                    Date = webResponse.Headers["Date"];
                }

                DateTime dateTime;

                dateTime = DateTime.Parse(Date);

                List<string> linkler = new List<string>();

                string icerik = "";
                string responseString = "";
                var encoding = ASCIIEncoding.UTF8;
                this.Invoke(new MethodInvoker(delegate ()
                {

                    lstDurum.TopIndex = lstDurum.Items.Add(url + " sayfası
internetten çekiliyor..");

                }));
                using (var responseStream =
webResponse.GetResponseStream())
```

```
                {
                    using (var streamReader = new
StreamReader(responseStream))
                    {
                        responseString = streamReader.ReadToEnd();
                    }
                }

                icerik = responseString;
                try
                {
                    HtmlAgilityPack.HtmlDocument htmldoc = new
HtmlAgilityPack.HtmlDocument();
                    htmldoc.LoadHtml(icerik);
                    linkler = addLinks(url, icerik);
                }
                catch (Exception ex)
                {
                    if (ex.Message.Contains("Object reference not set"))
                    {
                        var hw = new HtmlWeb();
                        var doc = hw.Load(url);
                        icerik = doc.DocumentNode.OuterHtml;
                        linkler = addLinks(url, icerik);
                    }
                }
                using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
                {

                    string deger = Database.addPage(url, dateTime,
responseString, linkler.Count, connection);

                    if (deger == "")
                    {

                        for (int i = 0; i < linkler.Count; i++)
                        {
                            if (url != linkler[i])
                            {
                                Database.addLink(url, linkler[i], connection);
                            }
                        }

                    }
                    else
                    {
                        throw new Exception(deger);
                    }
                    this.Invoke(new MethodInvoker(delegate ()
```

```
                {

                    lstDurum.TopIndex = lstDurum.Items.Add(url + " sayfası
kayıt edildi.");

                }));
            }


        }
        catch (Exception ex)
        {
            Debug.WriteLine("");
            Debug.WriteLine("### Form1.cs ###");
            Debug.WriteLine(ex.Message + ex.StackTrace);
            this.Invoke(new MethodInvoker(delegate ()
            {

                lstDurum.TopIndex = lstDurum.Items.Add(url + " -- Hata -- "
+ ex.Message);

            }));
            using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
            {
                Database.deleteLinkInLinkCollections(url, connection);
                Database.deleteLinkInLinks(url, connection);
                Database.deleteLinkInPageCollections(url, connection);
            }

            continue;
        }
        using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
        {
            Database.deleteLinkInLinkCollections(url, connection);
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine("");
        Debug.WriteLine("### Form1.cs ###");
        Debug.WriteLine(ex.Message + ex.StackTrace);
        using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
        {
            Database.deleteLinkInLinkCollections(url, connection);
            Database.deleteLinkInLinks(url, connection);
            Database.deleteLinkInPageCollections(url, connection);
        }
```

```
                    this.Invoke(new MethodInvoker(delegate ()
                        {
                            lstDurum.TopIndex = lstDurum.Items.Add(url + " -- Hata -- "
+ ex.Message);
                        }));
                    continue;
                }

            }
        }


        private List<string> addLinks(string url, string icerik)
        {
            List<string> linkler = new List<string>();
            if (!icerik.Contains("html"))
            {
                throw new Exception("");
            }
            HtmlAgilityPack.HtmlDocument htmldoc = new
HtmlAgilityPack.HtmlDocument();
            htmldoc.LoadHtml(icerik);
            HtmlNodeCollection colllection =
htmldoc.DocumentNode.SelectNodes("//a[@href]");
            if (colllection != null)
            {
                foreach (HtmlNode nodelink in colllection)
                {
                    HtmlAttribute att = nodelink.Attributes["href"];
                    var link = "";


                    if (att.Value == "" || att.Value.Contains("javascript") || att.Value ==
"/" || att.Value.Contains("#") || att.Value == "." || att.Value == "mailto:")
                    {
                        continue;
                    }

                    if (att.Value[0] == '.')
                    {
                        att.Value = att.Value.Remove(0, 1);
                    }



                    if (!att.Value.StartsWith("http"))
                    {
                        if (att.Value[0] == '.')
                        {
```

```
            att.Value = att.Value.Remove(0, 1);
        }
        if (att.Value[0] != '/')
        {
            link = "/" + att.Value;
        }
        else
        {
            link = att.Value;
        }
        link = url + link;
    }
    else
    {
        link = att.Value;
    }



    if (link[link.Length - 1] == '.')
    {
        link = link.Remove(link.Length - 1, 1);
    }

    if (link[link.Length - 1] == '/')
    {
        link = link.Remove(link.Length - 1, 1);
    }

    if (link.Substring(0, 10).Contains("http://"))
    {
        link = link.Remove(0, "http://".Length);
        link = link.Replace("//", "/");
        link = "http://" + link;
    }
    else if (link.Substring(0, 10).Contains("https://"))
    {
        link = link.Remove(0, "https://".Length);
        link = link.Replace("//", "/");
        link = "https://" + link;
    }
    char x = link.Last();
    if (x == '/')
    {
        link = link.Remove(link.Length - 1, 1);
    }

    if (link[link.Length - 1] == '.')
    {
```

```
                    link = link.Remove(link.Length - 1, 1);
                }

                if (link[link.Length - 1] == '/')
                {
                    link = link.Remove(link.Length - 1, 1);
                }

                if (url != link)
                {
                    if (!link.Contains(".jpg") && !link.Contains(".gif") &&
!link.Contains(".png") && !link.Contains(".jpeg") && !link.Contains(".pdf")
                    && !link.Contains(".doc") && !link.Contains(".Jpeg") &&
!link.Contains(".js") && !link.Contains(".css") && !link.Contains(".svg")
                    && !link.Contains(".mp4") && !link.Contains(".flv") &&
!link.Contains(".eot") && !link.Contains(".woff") && !link.Contains(".ttf")
                    && !link.Contains("'") && !link.Contains("<") &&
!link.Contains(".dtd") && !link.Contains(".JPG") && !link.Contains(".flv"))
                    {
                        if (link.Length < 200)
                        {
                            linkler.Add(link);
                        }
                    }

                }
            }
        }
        else
        {
            //html içerik script içerisine gömülmüş ilerde bunuda çıkartıcak bir
kod yazabilirsin.
        }
        foreach(string link in linkler)
        {
            link.Replace(" ", "");
        }
        linkler = linkler.Distinct().ToList();
        return linkler;

    }
    private void btnDuraklat_Click(object sender, EventArgs e)
    {
        Pause();
        btnDuraklat.Enabled = false;
        btnDevam.Enabled = true;
    }

    private void btnDevam_Click(object sender, EventArgs e)
    {
```

```
            Resume();
            btnDuraklat.Enabled = true;
            btnDevam.Enabled = false;
        }

        private void btnDurdur_Click(object sender, EventArgs e)
        {
            KillTheTasks();
            txtTaskCount.Enabled = true;
            btnBaslat.Enabled = true;
            btnDevam.Enabled = false;
            btnDuraklat.Enabled = false;
            btnDurdur.Enabled = false;
        }

        private void txtTaskCount_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
        (e.KeyChar != '.'))
            {
                e.Handled = true;
            }

            // only allow one decimal point
            if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))
            {
                e.Handled = true;
            }
        }

        private void kopyalaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(lstDurum.SelectedItem.ToString(),
TextDataFormat.Text);
        }

        private void lstDurum_MouseClick(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Right)
            {
                Point p = new Point(e.X, e.Y);
                int index = lstDurum.IndexFromPoint(p);
                if (index >=0)
                {
                    lstDurum.SelectedItem = lstDurum.Items[index];
                }
            }
        }
    }
```

```
public class TaskClass
{
   public TaskClass(Task task, CancellationTokenSource cancelTokenSource)
   {
      this.task = task;
      this.cancelTokenSource = cancelTokenSource;
   }
   public Task task { get; set; }
   public CancellationTokenSource cancelTokenSource { get; set; }
}
```

**Form1.Designer.cs**

```
namespace WebCrawler
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.components = new System.ComponentModel.Container();
         this.lstUrlList = new System.Windows.Forms.ListBox();
         this.label1 = new System.Windows.Forms.Label();
         this.lstDurum = new System.Windows.Forms.ListBox();
         this.label3 = new System.Windows.Forms.Label();
         this.btnBaslat = new System.Windows.Forms.Button();
         this.btnDuraklat = new System.Windows.Forms.Button();
```

```
            this.btnDevam = new System.Windows.Forms.Button();
            this.btnDurdur = new System.Windows.Forms.Button();
            this.label2 = new System.Windows.Forms.Label();
            this.txtTaskCount = new System.Windows.Forms.TextBox();
            this.contextMenuStrip1 = new
System.Windows.Forms.ContextMenuStrip(this.components);
            this.kopyalaToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.contextMenuStrip1.SuspendLayout();
            this.SuspendLayout();
            //
            // lstUrlList
            //
            this.lstUrlList.FormattingEnabled = true;
            this.lstUrlList.Items.AddRange(new object[] {
            "http://www.selcuk.edu.tr",
            "http://www.facebook.com",
            "http://www.twitter.com",
            "http://www.youtube.com",
            "http://www.yahoo.com",
            "http://www.instagram.com",
            "http://www.sabah.com",
            "http://www.haber7.com.tr",
            "http://www.sahibinden.com",
            "http://www.onedio.com",
            "http://www.hurriyet.com.tr",
            "http://www.milliyet.com.tr",
            "http://www.eksisozluk.com",
            "http://www.kizlarsoruyor.com",
            "http://www.hepsiburada.com",
            "http://www.n11.com",
            "http://www.gittigidiyor.com",
            "http://www.vogue.com.tr",
            "http://www.donanimhaber.com",
            "http://www.shiftdelete.net",
            "http://www.haberler.com"});
            this.lstUrlList.Location = new System.Drawing.Point(14, 34);
            this.lstUrlList.Name = "lstUrlList";
            this.lstUrlList.Size = new System.Drawing.Size(269, 251);
            this.lstUrlList.TabIndex = 0;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.label1.Location = new System.Drawing.Point(11, 18);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(83, 13);
```

```
            this.label1.TabIndex = 1;
            this.label1.Text = "URL LİSTESİ";
            //
            // lstDurum
            //
            this.lstDurum.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top
| System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.lstDurum.ContextMenuStrip = this.contextMenuStrip1;
            this.lstDurum.Font = new System.Drawing.Font("Microsoft Sans Serif",
10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.lstDurum.FormattingEnabled = true;
            this.lstDurum.ItemHeight = 16;
            this.lstDurum.Location = new System.Drawing.Point(14, 331);
            this.lstDurum.Name = "lstDurum";
            this.lstDurum.Size = new System.Drawing.Size(691, 164);
            this.lstDurum.TabIndex = 4;
            this.lstDurum.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.lstDurum_MouseClick);
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.label3.Location = new System.Drawing.Point(14, 312);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(53, 13);
            this.label3.TabIndex = 5;
            this.label3.Text = "DURUM";
            //
            // btnBaslat
            //
            this.btnBaslat.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnBaslat.Location = new System.Drawing.Point(465, 81);
            this.btnBaslat.Name = "btnBaslat";
            this.btnBaslat.Size = new System.Drawing.Size(154, 29);
            this.btnBaslat.TabIndex = 6;
            this.btnBaslat.Text = "BAŞLAT";
            this.btnBaslat.UseVisualStyleBackColor = true;
            this.btnBaslat.Click += new System.EventHandler(this.btnBaslat_Click);
            //
            // btnDuraklat
            //
```

```
            this.btnDuraklat.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDuraklat.Location = new System.Drawing.Point(465, 116);
            this.btnDuraklat.Name = "btnDuraklat";
            this.btnDuraklat.Size = new System.Drawing.Size(154, 29);
            this.btnDuraklat.TabIndex = 7;
            this.btnDuraklat.Text = "DURAKLAT";
            this.btnDuraklat.UseVisualStyleBackColor = true;
            this.btnDuraklat.Click                 +=                 new
System.EventHandler(this.btnDuraklat_Click);
            //
            // btnDevam
            //
            this.btnDevam.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F,      System.Drawing.FontStyle.Bold,      System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDevam.Location = new System.Drawing.Point(465, 151);
            this.btnDevam.Name = "btnDevam";
            this.btnDevam.Size = new System.Drawing.Size(154, 29);
            this.btnDevam.TabIndex = 8;
            this.btnDevam.Text = "DEVAM ETTİR";
            this.btnDevam.UseVisualStyleBackColor = true;
            this.btnDevam.Click                 +=                 new
System.EventHandler(this.btnDevam_Click);
            //
            // btnDurdur
            //
            this.btnDurdur.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F,      System.Drawing.FontStyle.Bold,      System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDurdur.Location = new System.Drawing.Point(465, 186);
            this.btnDurdur.Name = "btnDurdur";
            this.btnDurdur.Size = new System.Drawing.Size(154, 29);
            this.btnDurdur.TabIndex = 10;
            this.btnDurdur.Text = "DURDUR";
            this.btnDurdur.UseVisualStyleBackColor = true;
            this.btnDurdur.Click                 +=                 new
System.EventHandler(this.btnDurdur_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif",
10F,      System.Drawing.FontStyle.Bold,      System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.label2.Location = new System.Drawing.Point(462, 48);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(96, 17);
            this.label2.TabIndex = 11;
```

```
            this.label2.Text = "Task Sayısı:";
            //
            // txtTaskCount
            //
            this.txtTaskCount.Location = new System.Drawing.Point(565, 48);
            this.txtTaskCount.Name = "txtTaskCount";
            this.txtTaskCount.Size = new System.Drawing.Size(54, 20);
            this.txtTaskCount.TabIndex = 12;
            this.txtTaskCount.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.txtTaskCount_KeyPress);
            //
            // contextMenuStrip1
            //
            this.contextMenuStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.kopyalaToolStripMenuItem});
            this.contextMenuStrip1.Name = "contextMenuStrip1";
            this.contextMenuStrip1.Size = new System.Drawing.Size(117, 26);
            //
            // kopyalaToolStripMenuItem
            //
            this.kopyalaToolStripMenuItem.Name = "kopyalaToolStripMenuItem";
            this.kopyalaToolStripMenuItem.Size = new System.Drawing.Size(180,
22);
            this.kopyalaToolStripMenuItem.Text = "Kopyala";
            this.kopyalaToolStripMenuItem.Click += new
System.EventHandler(this.kopyalaToolStripMenuItem_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(716, 503);
            this.Controls.Add(this.txtTaskCount);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.btnDurdur);
            this.Controls.Add(this.btnDevam);
            this.Controls.Add(this.btnDuraklat);
            this.Controls.Add(this.btnBaslat);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.lstDurum);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.lstUrlList);
            this.Name = "Form1";
            this.Text = "Multi Web Crawler Uygulaması";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.contextMenuStrip1.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();
```

```
            }

            #endregion

            private System.Windows.Forms.ListBox lstUrlList;
            private System.Windows.Forms.Label label1;
            private System.Windows.Forms.ListBox lstDurum;
            private System.Windows.Forms.Label label3;
            private System.Windows.Forms.Button btnBaslat;
            private System.Windows.Forms.Button btnDuraklat;
            private System.Windows.Forms.Button btnDevam;
            private System.Windows.Forms.Button btnDurdur;
            private System.Windows.Forms.Label label2;
            private System.Windows.Forms.TextBox txtTaskCount;
            private System.Windows.Forms.ContextMenuStrip contextMenuStrip1;
            private System.Windows.Forms.ToolStripMenuItem
kopyalaToolStripMenuItem;
        }
    }
```

**Database.cs**

```
    using Microsoft.TeamFoundation.WorkItemTracking.Controls;
    using MongoDB.Bson;
    using MongoDB.Driver;
    using MySql.Data.MySqlClient;
    using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Data.SqlClient;
    using System.Diagnostics;
    using System.IO;
    using System.Linq;
    using System.Text;
    using System.Text.RegularExpressions;
    using System.Threading.Tasks;

    namespace WebCrawler
    {
        public class LinkCollection
        {
            public string Url { get; set; }
        }
        public class Database
        {
            public static string connectionString =
@"server=localhost;userid=root;password=karlik;database=search_engine;pooling=true
;max pool size=10000;";
            public static DataTable get1000RecordInLinkCollection()
            {
```

```
            MySqlConnection connection = new MySqlConnection(connectionString);
            if (connection.State != ConnectionState.Open)
               connection.Open();
            string stm = "set net_write_timeout=99999; set net_read_timeout=99999;" + "SELECT * FROM linkcollection Where Limit 1000 ";
            MySqlDataAdapter dataAdapter = new MySqlDataAdapter();
            dataAdapter.SelectCommand = new MySqlCommand(stm, connection);
            DataTable table = new DataTable();
            dataAdapter.Fill(table);
            connection.Close();
            return table;
         }

         public static void addLink(string url, string link, MySqlConnection connection)
         {
            try
            {

               if (link != null)
               {
                  bool varmi = Database.getRecordInLinks(url, link, connection);
                  if (varmi == false)
                  {
                     using (MySqlCommand cmd = new MySqlCommand())
                     {
                        cmd.Connection = connection;
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                           connection.Open();
                        cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "INSERT INTO links(url,link) VALUES(@url, @link);";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@url", url);
                        cmd.Parameters.AddWithValue("@link", link);
                        cmd.ExecuteNonQuery();
                        connection.Close();
                     }
                  }

                  varmi = Database.getRecordInLinkCollection(link, connection);
                  if (varmi == false)
                  {
                     using (MySqlCommand cmd = new MySqlCommand())
                     {
                        cmd.Connection = connection;
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                           connection.Open();
```

```
                        cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "INSERT INTO linkcollection (url) VALUES (@link);";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@link", link);
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }
                }
            }
            else
            {
                bool varmi = Database.getRecordInLinkCollection(link, connection);
                if (varmi == false)
                {
                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.Connection = connection;
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "INSERT INTO linkcollection (url) VALUES (@link);";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@link", link);
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }
                }
            }

        }
        catch (Exception ex)
        {
            Debug.WriteLine("");
            Debug.WriteLine("### addLink ### " + "Url: " + url + " Link: " + link + " Connection: " + connection.State.ToString());
            Debug.WriteLine(ex.Message);
            connection.Close();
        }

    }

    private static bool getRecordInLinks(string url, string link, MySqlConnection connection)
    {
        bool varmi = false;
        {
            try
            {
```

```
                string stm = "SELECT * FROM links where url=@url and
link=@link;";

                using (MySqlCommand cmd = new MySqlCommand("set
net_write_timeout=99999; set net_read_timeout=99999;" + stm, connection))
                {
                   cmd.CommandTimeout = 60000;
                   cmd.Parameters.AddWithValue("@url", url);
                   cmd.Parameters.AddWithValue("@link", link);

                   if (connection.State != ConnectionState.Open)
                      connection.Open();
                   using (MySqlDataReader reader = cmd.ExecuteReader())
                   {
                      while (reader.HasRows)
                      {
                         varmi = true;
                         break;
                      }
                      reader.Close();
                   }
                   connection.Close();
                }

             }
             catch (Exception ex)
             {
                Debug.WriteLine("");
                Debug.WriteLine("### getRecordInLinks ###" + "Url: " + url + "
Link: " + link + " Connection: " + connection.State.ToString());
                Debug.WriteLine(ex.Message);
                connection.Close();
             }
          }

          return varmi;
       }

       public static string addPage(string url, DateTime? tarih, string htmlIcerik,
int linkCount, MySqlConnection connection)
       {
          double pagerank = 0.15f;
          string titleString = "", icerik = "";
          DateTime? indekslemeTarihi = null, kayitTarihi = tarih;
          bool indekslemeYapiliyormu = false;
          string str = "";
          int counted = 0;

          try
          {
```

```
                DataTable table = new DataTable();
                using (MySqlCommand cmd = new MySqlCommand())
                {
                   cmd.Connection = connection;
                   cmd.CommandTimeout = 60000;
                   if (connection.State != ConnectionState.Open)
                      connection.Open();
                   string stm = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "SELECT * FROM pagecollection where url=@url;";
                   MySqlDataAdapter dataAdapter = new MySqlDataAdapter();
                   cmd.CommandText = stm;
                   cmd.Prepare();
                   cmd.Parameters.AddWithValue("@url", url);
                   dataAdapter.SelectCommand = cmd;
                   dataAdapter.Fill(table);
                   connection.Close();
                }
                using (MySqlCommand cmd = new MySqlCommand())
                {

                   if (table.Rows.Count != 0)
                   {
                      counted = int.Parse(table.Rows[0]["counted"].ToString());
                      cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "UPDATE pagecollection SET title = @title,htmlicerik =
@htmlicerik,icerik = @icerik,pagerank = @pagerank, kayittarihi =
@kayittarihi,indekslemetarihi = @indekslemetarihi,indekslemeyapiliyormu =
@indekslemeyapiliyormu, linkCount = @linkCount, counted = @counted WHERE
url=@url;";
                   }
                   else
                   {
                      cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "INSERT INTO
pagecollection(url,title,htmlicerik,icerik,pagerank,kayittarihi,indekslemetarihi,indeksle
meyapiliyormu,linkcount,counted)
VALUES(@url,@title,@htmlicerik,@icerik,@pagerank,@kayittarihi,@indekslemetari
hi,@indekslemeyapiliyormu, @linkCount, @counted)";
                   }

                   HtmlAgilityPack.HtmlDocument doc = new
HtmlAgilityPack.HtmlDocument();
                   doc.LoadHtml(htmlIcerik);
                   var title = doc.DocumentNode.SelectSingleNode("//title");
                   icerik = HtmlFilter.ConvertToPlainText(htmlIcerik);

                   icerik = Regex.Replace(icerik, @"\s+", " ");
                   icerik = Regex.Replace(icerik, @"\t+", " ");
                   icerik = Regex.Replace(icerik, @"\r+", " ");
```

```
                    //htmlIcerik = Zip(htmlIcerik);
                    //icerik = Zip(icerik);

                    titleString = "";

                    if (title != null)
                    {
                        titleString = title.InnerText;
                    }

                    if (table.Rows.Count != 0)
                    {
                        pagerank = Double.Parse(table.Rows[0]["pagerank"].ToString());
                        Debug.WriteLine(pagerank);
                    }
                    else
                    {
                        pagerank = 0.15f;
                        Debug.WriteLine(pagerank);
                    }


                    cmd.Connection = connection;
                    cmd.CommandTimeout = 60000;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@url", url);
                    cmd.Parameters.AddWithValue("@title", titleString);
                    cmd.Parameters.AddWithValue("@htmlicerik", htmlIcerik);
                    cmd.Parameters.AddWithValue("@icerik", icerik);
                    cmd.Parameters.AddWithValue("@pagerank", pagerank);
                    cmd.Parameters.AddWithValue("@kayittarihi", kayitTarihi);
                    cmd.Parameters.AddWithValue("@indekslemetarihi",
indekslemeTarihi);
                    cmd.Parameters.AddWithValue("@indekslemeyapiliyormu",
indekslemeYapiliyormu);
                    cmd.Parameters.AddWithValue("@linkCount", linkCount);
                    cmd.Parameters.AddWithValue("@counted", counted);
                    cmd.ExecuteNonQuery();
                    connection.Close();

                    str = "";
                }
            }
            catch (Exception ex)
            {
                connection.Close();
                try
                {
```

```
                    Debug.WriteLine("");
                    Debug.WriteLine("### addPage ###" + "Url: " + url + " Title: " +
titleString.Substring(0, 10) + " HtmlIcerik: " + htmlIcerik.Substring(0, 10) + " Icerik: "
+ icerik.Substring(0, 10) + " PageRank: " + pagerank +
                        " KayitTarihi: " + kayitTarihi + " IndekslemeTarihi: " +
indekslemeTarihi + " IndekslemeYapiliyormu: " + indekslemeYapiliyormu + "
LinkCount: " + linkCount + " Counted: " + counted + " Connection: " +
connection.State.ToString());
                    Debug.WriteLine(ex.Message);
                    str = ex.Message + ex.StackTrace;
                }
                catch (Exception ex2)
                {
                    Debug.WriteLine("### addPage ###" + "Url: " + url + "
Connection: " + connection.State.ToString());
                    Debug.WriteLine(ex.Message + ex2.Message);
                }


            }

            return str;
        }

        public static bool getRecordInPageCollection(string url, MySqlConnection
connection)
        {
            bool varmi = false;

            try
            {
                string stm = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "SELECT * FROM pageCollection where url=@url;";
                using (MySqlCommand cmd = new MySqlCommand(stm,
connection))
                {
                    cmd.CommandTimeout = 60000;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@url", url);

                    using (MySqlDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.HasRows)
                        {
                            varmi = true;
                            break;
                        }
                        reader.Close();
```

```
                }
                connection.Close();
            }

        }
        catch (Exception ex)
        {
            connection.Close();
            Debug.WriteLine("");
            Debug.WriteLine("### getRecordInPageCollection ###" + "Url: " + url);
            Debug.WriteLine(ex.Message);
        }

        return varmi;
    }


    public static bool getRecordInLinkCollection(string url, MySqlConnection connection)
    {

        bool varmi = false;

        try
        {
            string stm = "set net_write_timeout=99999; set net_read_timeout=99999;" + "SELECT * FROM linkCollection where url=@url;";
            using (MySqlCommand cmd = new MySqlCommand(stm, connection))
            {
                cmd.CommandTimeout = 60000;
                if (connection.State != ConnectionState.Open)
                    connection.Open();
                cmd.Prepare();
                cmd.Parameters.AddWithValue("@url", url);

                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.HasRows)
                    {
                        varmi = true;
                        break;
                    }
                    reader.Close();
                }
                connection.Close();
            }

        }
```

```
                catch (Exception ex)
                {
                    connection.Close();
                    Debug.WriteLine("");
                    Debug.WriteLine("### getRecordInLinkCollection ###" + "Url: " +
url);
                    Debug.WriteLine(ex.Message);
                }

                return varmi;
            }


            public static void deleteDatasInCollections(MySqlConnection connection)
            {
                try
                {
                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.Connection = connection;
                        cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from pagecollection;";
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }
                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.Connection = connection;
                        cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from linkcollection;";
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }

                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.CommandTimeout = 60000;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.Connection = connection;
                        cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from links;";
                        cmd.ExecuteNonQuery();
                        connection.Close();
```

```
                }

            }
            catch (Exception ex)
            {
                connection.Close();
                Debug.WriteLine("");
                Debug.WriteLine("### deleteDatasInCollections ###");
                Debug.WriteLine(ex.Message);
            }
        }

        public static void deleteLinkInLinkCollections(string url, MySqlConnection connection)
        {
            try
            {
                using (MySqlCommand cmd = new MySqlCommand())
                {
                    cmd.CommandTimeout = 60000;
                    cmd.Connection = connection;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "delete from linkcollection where url=@url;";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@url", url);
                    cmd.ExecuteNonQuery();
                    connection.Close();
                }
            }
            catch (Exception ex)
            {
                connection.Close();
                Debug.WriteLine("");
                Debug.WriteLine("### deleteLinkInLinkCollections ###" + "Url: " + url);
                Debug.WriteLine(ex.Message);
            }

        }


        public static void deleteLinkInPageCollections(string url, MySqlConnection connection)
        {
            try
            {
                using (MySqlCommand cmd = new MySqlCommand())
                {
```

```
                    cmd.CommandTimeout = 60000;
                    cmd.Connection = connection;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from pagecollection where url=@url;";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@url", url);
                    cmd.ExecuteNonQuery();
                    connection.Close();
                }
            }
            catch (Exception ex)
            {
                connection.Close();
                Debug.WriteLine("");
                Debug.WriteLine("### deleteLinkInPageCollections ###" + "Url: " +
url);
                Debug.WriteLine(ex.Message);
            }

        }


        public static void deleteLinkInLinks(string url, MySqlConnection
connection)
        {
            try
            {
                using (MySqlCommand cmd = new MySqlCommand())
                {
                    cmd.CommandTimeout = 60000;
                    cmd.Connection = connection;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from links where url=@url;";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@url", url);
                    cmd.ExecuteNonQuery();
                    connection.Close();
                }
                using (MySqlCommand cmd = new MySqlCommand())
                {
                    cmd.CommandTimeout = 60000;
                    cmd.Connection = connection;
                    if (connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "delete from links where link=@url;";
```

```
                cmd.Prepare();
                cmd.Parameters.AddWithValue("@url", url);
                cmd.ExecuteNonQuery();
                connection.Close();
            }
        }
        catch (Exception ex)
        {
            connection.Close();
            Debug.WriteLine("");
            Debug.WriteLine("### deleteLinkInLinks ###" + "Url: " + url);
            Debug.WriteLine(ex.Message);
        }


    }


    //public static string Zip(string text)
    //{
    //    byte[] buffer = System.Text.Encoding.Unicode.GetBytes(text);
    //    MemoryStream ms = new MemoryStream();
    //        using (System.IO.Compression.GZipStream zip = new
System.IO.Compression.GZipStream(ms,
System.IO.Compression.CompressionMode.Compress, true))
    //    {
    //        zip.Write(buffer, 0, buffer.Length);
    //    }

    //    ms.Position = 0;
    //    MemoryStream outStream = new MemoryStream();

    //    byte[] compressed = new byte[ms.Length];
    //    ms.Read(compressed, 0, compressed.Length);

    //    byte[] gzBuffer = new byte[compressed.Length + 4];
    //        System.Buffer.BlockCopy(compressed, 0, gzBuffer, 4,
compressed.Length);
    //        System.Buffer.BlockCopy(BitConverter.GetBytes(buffer.Length), 0,
gzBuffer, 0, 4);
    //    return Convert.ToBase64String(gzBuffer);
    //}

    //public static string UnZip(string compressedText)
    //{
    //    byte[] gzBuffer = Convert.FromBase64String(compressedText);
    //    using (MemoryStream ms = new MemoryStream())
    //    {
    //        int msgLength = BitConverter.ToInt32(gzBuffer, 0);
    //        ms.Write(gzBuffer, 4, gzBuffer.Length - 4);
```

```
            //      byte[] buffer = new byte[msgLength];

            //      ms.Position = 0;
            //                  using (System.IO.Compression.GZipStream zip = new
System.IO.Compression.GZipStream(ms,
System.IO.Compression.CompressionMode.Decompress))
            //      {
            //          zip.Read(buffer, 0, buffer.Length);
            //      }

            //                  return System.Text.Encoding.Unicode.GetString(buffer, 0,
buffer.Length);
            //  }
            //}


        }
    }
```

## EK B - WEBINDEXER UYGULAMASI

### Form1.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Collections.Concurrent;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Security.Permissions;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace WebIndexer
{
   public partial class Form1 : Form
   {
      List<TaskClass> taskListesi = new List<TaskClass>();
      ManualResetEvent mrse = new ManualResetEvent(false);
      ConcurrentQueue<DataRow> kuyruk = new
ConcurrentQueue<DataRow>();
      bool kuyrukBosmu;
      int taskCount = 0;
      public Form1()
      {
         InitializeComponent();
         kuyrukBosmu = true;
         btnBaslat.Enabled = false;
         btnDevam.Enabled = false;
         btnDuraklat.Enabled = false;
         btnDurdur.Enabled = false;

         KuyrukDoldur();

         btnBaslat.Enabled = true;
      }

      public void KuyrukDoldur()
      {
         if (kuyrukBosmu == false)
```

```
            {
                return;
            }

            DataTable dt = Database.KuyrukDoldur();
            foreach (DataRow dr in dt.Rows)
            {
                kuyruk.Enqueue(dr);
            }
            kuyrukBosmu = false;

        }

        public void BasaDon()
        {

            this.Invoke(new MethodInvoker(delegate ()
            {
                btnDuraklat.Enabled = false;
                btnDevam.Enabled = false;
                btnDurdur.Enabled = false;
                btnBaslat.Enabled = true;

            }));


            this.Invoke(new MethodInvoker(delegate ()
            {

                lstDurum.HorizontalScrollbar = true;
                lstDurum.Items.Clear();

            }));
        }

        public void Resume()
        {
            mrse.Set();
        }

        public void Pause()
        {
            mrse.Reset();
        }

        [SecurityPermissionAttribute(SecurityAction.Demand, ControlThread =
true)]
        private void KillTheTasks()
        {
```

```
            Pause();
            for (int i = 0; i < taskListesi.Count; i++)
            {
                TaskClass t = taskListesi[i];

                t.cancelTokenSource.Cancel();

                using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
                {
                    Database.updatePage(t.pageId, null, false, connection);
                }


            }

            taskListesi = new List<TaskClass>();
        }

        private void btnBaslat_Click(object sender, EventArgs e)
        {

            if (txtTaskCount.Text != "")
                taskCount = int.Parse(txtTaskCount.Text);

            if (taskCount > 10 || taskCount < 1)
            {
                MessageBox.Show("1 ile 10 aralığında bir değer giriniz.");
                txtTaskCount.Text = "";
                return;
            }


            CancellationTokenSource cts = new CancellationTokenSource();
            CancellationToken ct = cts.Token;
            Task task = new Task(taskStart, ct);

            task.Start();

            taskListesi.Add(new TaskClass(task, cts, task.Id, 0));
            txtTaskCount.Enabled = false;
            btnBaslat.Enabled = false;
            btnDuraklat.Enabled = true;
            btnDurdur.Enabled = true;
            Resume();
        }

        private void taskStart()
        {
            taskListesi = new List<TaskClass>();
```

```
            for (int i = 0; i < taskCount; i++)
            {
                mrse.WaitOne();
                CancellationTokenSource cts = new CancellationTokenSource();
                CancellationToken ct = cts.Token;
                Task task = new Task(index, ct);

                task.Start();
                taskListesi.Add(new TaskClass(task, cts, task.Id, 0));
                Thread.Sleep(1500);

            }
        }

        private void index()
        {

            while (true)
            {
                mrse.WaitOne();
                try
                {
                    if (kuyruk.Count > 0)
                    {
                        DataRow dr = null;
                        kuyruk.TryDequeue(out dr);
                        using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
                        {
                            Database.updatePage((int)dr["id"], null, true, connection);
                        }

                        this.Invoke(new MethodInvoker(delegate ()
                        {
                            lstDurum.TopIndex = lstDurum.Items.Add(dr["url"].ToString()
+ " sayfası indexleniyor..");
                        }));

                        string icerik = dr["icerik"].ToString();
                        int id = (int)dr["id"];
                        for (int i = 0; i < taskListesi.Count; i++)
                        {
                            if (Task.CurrentId == taskListesi[i].taskId)
                            {
                                taskListesi[i].pageId = id;
                            }
                        }

                        icerik = Regex.Replace(icerik, @"[^\w\s]", "");
                        string[] dizi = icerik.Split(' ');
```

```
                    List<string> list = dizi.ToList();
                    list = list.ConvertAll(d => d.ToLower());
                    list.RemoveAll(item => item.ToLower() == "in");
list.RemoveAll(item => item.ToLower() == "a"); list.RemoveAll(item => item.ToLower() == "or"); list.RemoveAll(item => item.ToLower() == "and");
                    list.RemoveAll(item => item.ToLower() == "b");
list.RemoveAll(item => item.ToLower() == "c"); list.RemoveAll(item => item.ToLower() == "c"); list.RemoveAll(item => item.ToLower() == "e");
                    list.RemoveAll(item => item.ToLower() == "f");
list.RemoveAll(item => item.ToLower() == "g"); list.RemoveAll(item => item.ToLower() == "h"); list.RemoveAll(item => item.ToLower() == "ı");
                    list.RemoveAll(item => item.ToLower() == "i");
list.RemoveAll(item => item.ToLower() == "j"); list.RemoveAll(item => item.ToLower() == "k"); list.RemoveAll(item => item.ToLower() == "l");
                    list.RemoveAll(item => item.ToLower() == "m");
list.RemoveAll(item => item.ToLower() == "n"); list.RemoveAll(item => item.ToLower() == "o"); list.RemoveAll(item => item.ToLower() == "ö");
                    list.RemoveAll(item => item.ToLower() == "p");
list.RemoveAll(item => item.ToLower() == "s"); list.RemoveAll(item => item.ToLower() == "ü"); list.RemoveAll(item => item.ToLower() == "y");
                    list.RemoveAll(item => item.ToLower() == "q");
list.RemoveAll(item => item.ToLower() == "t"); list.RemoveAll(item => item.ToLower() == "v"); list.RemoveAll(item => item.ToLower() == "z");
                    list.RemoveAll(item => item.ToLower() == "r");
list.RemoveAll(item => item.ToLower() == "u"); list.RemoveAll(item => item.ToLower() == "x"); list.RemoveAll(item => item.ToLower() == "w");
                    list.RemoveAll(item => item.ToLower() == "veya");
list.RemoveAll(item => item.ToLower() == "ve"); list.RemoveAll(item => item.ToLower() == "");

                    list = list.ConvertAll(d => d.ToLower());
                    dizi = list.ToArray();
                    var counts = new Dictionary<string, int>();

                    foreach (string value in dizi)
                    {
                       if (counts.ContainsKey(value))
                          counts[value] = counts[value] + 1;
                       else
                          counts.Add(value, 1);
                    }
                    using (MySqlConnection connection = new
MySqlConnection(Database.connectionString))
                    {
                       Database.addToken(id, counts, connection);

                       Database.updatePage(id, DateTime.Now, false, connection);
                    }

                    this.Invoke(new MethodInvoker(delegate ()
```

```
                        {
                            lstDurum.TopIndex = lstDurum.Items.Add(dr["url"].ToString()
+ " sayfasının indekslenme işlemi tamamlandı.");
                        }));
                        Thread.Sleep(1000);

                    }
                    else
                    {
                        if (kuyrukBosmu == true)
                            continue;

                        Pause();
                        kuyrukBosmu = true;
                        KuyrukDoldur();
                        kuyrukBosmu = false;
                        Resume();
                    }
                }
                catch (Exception ex)
                {
                    this.Invoke(new MethodInvoker(delegate ()
                    {
                        if (!ex.Message.Contains("Thread was being aborted"))
                        {

                            lstDurum.TopIndex = lstDurum.Items.Add(" -- Hata -- " +
ex.Message);
                        }

                    }));
                    continue;
                }

            }
        }
        private void btnDuraklat_Click(object sender, EventArgs e)
        {
            Pause();
            btnDuraklat.Enabled = false;
            btnDevam.Enabled = true;
        }

        private void btnDevam_Click(object sender, EventArgs e)
        {
            Resume();
            btnDuraklat.Enabled = true;
            btnDevam.Enabled = false;
        }
```

```
        private void btnDurdur_Click(object sender, EventArgs e)
        {

            KillTheTasks();
            btnBaslat.Enabled = true;
            txtTaskCount.Enabled = true;
            btnDevam.Enabled = false;
            btnDuraklat.Enabled = false;
            btnDurdur.Enabled = false;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            var cts = new CancellationTokenSource();
            CancellationToken ct = cts.Token;
            Task ts = new Task(BasaDon, ct);
            ts.Start();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            e.Cancel = true;
            Pause();
            label3.Text = "Form Kapanıyor Lütfen Bekleyiniz..";
            KillTheTasks();
            e.Cancel = false;
        }

        private void txtTaskCount_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
        (e.KeyChar != '.'))
            {
                e.Handled = true;
            }

            // only allow one decimal point
            if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))
            {
                e.Handled = true;
            }

        }

        private void kopyalaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(lstDurum.SelectedItem.ToString(),
TextDataFormat.Text);
        }
    }
```

```
        public class TaskClass
        {
            public TaskClass(Task task, CancellationTokenSource cancelTokenSource,
int taskId, int pageId)
            {
                this.task = task;
                this.cancelTokenSource = cancelTokenSource;
                this.taskId = taskId;
                this.pageId = pageId;
            }
            public Task task { get; set; }
            public int pageId { get; set; }
            public int taskId { get; set; }
            public CancellationTokenSource cancelTokenSource { get; set; }
        }

    }
```

**Form1.Designer.cs**

```
    namespace WebIndexer
    {
        partial class Form1
        {
            /// <summary>
            /// Required designer variable.
            /// </summary>
            private System.ComponentModel.IContainer components = null;

            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
            protected override void Dispose(bool disposing)
            {
                if (disposing && (components != null))
                {
                    components.Dispose();
                }
                base.Dispose(disposing);
            }

            #region Windows Form Designer generated code

            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
```

```
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.lstDurum = new System.Windows.Forms.ListBox();
            this.label3 = new System.Windows.Forms.Label();
            this.btnBaslat = new System.Windows.Forms.Button();
            this.btnDuraklat = new System.Windows.Forms.Button();
            this.btnDevam = new System.Windows.Forms.Button();
            this.btnDurdur = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.txtTaskCount = new System.Windows.Forms.TextBox();
            this.contextMenuStrip1 = new
System.Windows.Forms.ContextMenuStrip(this.components);
            this.kopyalaToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.contextMenuStrip1.SuspendLayout();
            this.SuspendLayout();
            //
            // lstDurum
            //
            this.lstDurum.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top
| System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.lstDurum.ContextMenuStrip = this.contextMenuStrip1;
            this.lstDurum.Font = new System.Drawing.Font("Arial", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.lstDurum.ForeColor = System.Drawing.Color.DarkSlateGray;
            this.lstDurum.FormattingEnabled = true;
            this.lstDurum.ItemHeight = 16;
            this.lstDurum.Location = new System.Drawing.Point(14, 81);
            this.lstDurum.Name = "lstDurum";
            this.lstDurum.Size = new System.Drawing.Size(691, 132);
            this.lstDurum.TabIndex = 4;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.label3.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.label3.Location = new System.Drawing.Point(14, 62);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(61, 13);
            this.label3.TabIndex = 5;
            this.label3.Text = "DURUM :";
```

```
            //
            // btnBaslat
            //
            this.btnBaslat.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnBaslat.Location = new System.Drawing.Point(215, 22);
            this.btnBaslat.Name = "btnBaslat";
            this.btnBaslat.Size = new System.Drawing.Size(114, 29);
            this.btnBaslat.TabIndex = 6;
            this.btnBaslat.Text = "BAŞLAT";
            this.btnBaslat.UseVisualStyleBackColor = true;
            this.btnBaslat.Click += new System.EventHandler(this.btnBaslat_Click);
            //
            // btnDuraklat
            //
            this.btnDuraklat.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDuraklat.Location = new System.Drawing.Point(335, 22);
            this.btnDuraklat.Name = "btnDuraklat";
            this.btnDuraklat.Size = new System.Drawing.Size(114, 29);
            this.btnDuraklat.TabIndex = 7;
            this.btnDuraklat.Text = "DURAKLAT";
            this.btnDuraklat.UseVisualStyleBackColor = true;
            this.btnDuraklat.Click += new
System.EventHandler(this.btnDuraklat_Click);
            //
            // btnDevam
            //
            this.btnDevam.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDevam.Location = new System.Drawing.Point(455, 22);
            this.btnDevam.Name = "btnDevam";
            this.btnDevam.Size = new System.Drawing.Size(114, 29);
            this.btnDevam.TabIndex = 8;
            this.btnDevam.Text = "DEVAM ETTİR";
            this.btnDevam.UseVisualStyleBackColor = true;
            this.btnDevam.Click += new
System.EventHandler(this.btnDevam_Click);
            //
            // btnDurdur
            //
            this.btnDurdur.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.btnDurdur.Location = new System.Drawing.Point(575, 22);
            this.btnDurdur.Name = "btnDurdur";
            this.btnDurdur.Size = new System.Drawing.Size(114, 29);
```

```
            this.btnDurdur.TabIndex = 10;
            this.btnDurdur.Text = "DURDUR";
            this.btnDurdur.UseVisualStyleBackColor = true;
            this.btnDurdur.Click += new
System.EventHandler(this.btnDurdur_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
            this.label1.Location = new System.Drawing.Point(14, 30);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(76, 13);
            this.label1.TabIndex = 11;
            this.label1.Text = "Task Count:";
            //
            // txtTaskCount
            //
            this.txtTaskCount.Location = new System.Drawing.Point(96, 27);
            this.txtTaskCount.Name = "txtTaskCount";
            this.txtTaskCount.Size = new System.Drawing.Size(38, 20);
            this.txtTaskCount.TabIndex = 12;
            this.txtTaskCount.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.txtTaskCount_KeyPress);
            //
            // contextMenuStrip1
            //
            this.contextMenuStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.kopyalaToolStripMenuItem});
            this.contextMenuStrip1.Name = "contextMenuStrip1";
            this.contextMenuStrip1.Size = new System.Drawing.Size(117, 26);
            //
            // kopyalaToolStripMenuItem
            //
            this.kopyalaToolStripMenuItem.Name = "kopyalaToolStripMenuItem";
            this.kopyalaToolStripMenuItem.Size = new System.Drawing.Size(180,
22);
            this.kopyalaToolStripMenuItem.Text = "Kopyala";
            this.kopyalaToolStripMenuItem.Click += new
System.EventHandler(this.kopyalaToolStripMenuItem_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(715, 222);
            this.Controls.Add(this.txtTaskCount);
```

```
            this.Controls.Add(this.label1);
            this.Controls.Add(this.btnDurdur);
            this.Controls.Add(this.btnDevam);
            this.Controls.Add(this.btnDuraklat);
            this.Controls.Add(this.btnBaslat);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.lstDurum);
            this.Name = "Form1";
            this.Text = "Web Indexer";
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
            this.Load += new System.EventHandler(this.Form1_Load);
            this.contextMenuStrip1.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion
        private System.Windows.Forms.ListBox lstDurum;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button btnBaslat;
        private System.Windows.Forms.Button btnDuraklat;
        private System.Windows.Forms.Button btnDevam;
        private System.Windows.Forms.Button btnDurdur;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox txtTaskCount;
        private System.Windows.Forms.ContextMenuStrip contextMenuStrip1;
        private System.Windows.Forms.ToolStripMenuItem
kopyalaToolStripMenuItem;
    }
}
```

**Database.cs**

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Xml;
using static System.Net.Mime.MediaTypeNames;

namespace WebIndexer
{
   public class LinkCollection
```

```
{
    public string Url { get; set; }
}
public class Database
{

    public static string connectionString = @"server=localhost;userid=root;password=karlik;database=search_engine;pooling=true;max pool size=1000000; Allow User Variables=True;";

    public static DataTable KuyrukDoldur()
    {
        DataTable dt = new DataTable();

        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            try
            {
                string stm = "set net_write_timeout=99999; set net_read_timeout=99999;" + "SELECT * FROM search_engine.pagecollection USE INDEX (NewIndex) where (kayitTarihi > indekslemeTarihi OR indekslemeTarihi is null) AND indekslemeyapiliyormu = 0 Limit 100;";
                MySqlDataAdapter dataAdapter = new MySqlDataAdapter();
                MySqlCommand cmd = new MySqlCommand(stm, connection);
                cmd.CommandTimeout = 600;
                dataAdapter.SelectCommand = cmd;
                dataAdapter.Fill(dt);
            }
            catch (Exception ex)
            {
                Debug.WriteLine("### KuyrukDoldur ###" + ex.Message + ex.StackTrace);
            }
        }
        return dt;
    }

    public static void updatePage(int id, DateTime? indekslemeTarihi, bool indekslemeYapiliyormu, MySqlConnection connection)
    {
        try
        {
            using (MySqlCommand cmd = new MySqlCommand())
            {
                cmd.Connection = connection;
            cmd.CommandTimeout = 600;
            if (connection.State != ConnectionState.Open)
                connection.Open();
```

```
                    cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "UPDATE pagecollection SET indekslemetarihi =
@indekslemetarihi,indekslemeyapiliyormu = @indekslemeyapiliyormu WHERE
id=@id;";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@id", id);
                    cmd.Parameters.AddWithValue("@indekslemetarihi",
indekslemeTarihi);
                    cmd.Parameters.AddWithValue("@indekslemeyapiliyormu",
indekslemeYapiliyormu);
                    cmd.ExecuteNonQuery();
                    connection.Close();
                }

            }
            catch (Exception ex)
            {
                Debug.WriteLine("### updatePage ###" + " PageId: " + id + " Hata:
" + ex.Message);
            }
            finally
            {
                if (connection != null)
                    connection.Close();
            }

        }

        public static string getToken(string token, MySqlConnection connection)
        {
            string array = "";
            try
            {
                string stm = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "SELECT * FROM search_engine.bigindex where
token=@token;";
                using (MySqlCommand cmd = new MySqlCommand(stm,
connection))
                {
                    cmd.CommandTimeout = 600;
                    if(connection.State != ConnectionState.Open)
                        connection.Open();
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@token", token);
                    using (MySqlDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            array = reader["array"].ToString();
                            break;
```

```
                }
                reader.Close();
            }
            connection.Close();
        }

    }
    catch (Exception ex)
    {
        connection.Close();
        Debug.WriteLine("");
        Debug.WriteLine("### getToken ###" + "Token: " + token);
        Debug.WriteLine(ex.Message);
        array = "error";
    }

    return array;
}

public static void addToken(int docId, Dictionary<string, int> liste, MySqlConnection connection)
{
    string token = "";
    string etArray = "";
    try
    {
        for (int i = 0; i < liste.Count; i++)
        {
            token = liste.Keys.ElementAt(i);
            int frekans = liste.Values.ElementAt(i);

            string array = getToken(token,connection);

            if (array != "" && array != "error")
            {
                DataSet ds = new DataSet();
                ds.ReadXml(XmlReader.Create(new StringReader(array)));
                DataTable dt = ds.Tables[0];
                if (dt != null)
                {
                    bool docIdVarmi = false;
                    string id = docId.ToString();
                    foreach (DataRow row in dt.Rows)
                    {
                        if (row["docId"].ToString() == id)
                        {
                            docIdVarmi = true;
                            row["frekans"] = frekans;

                            dt.AcceptChanges();
```

```
                        ds.AcceptChanges();


                        using (MySqlCommand cmd = new MySqlCommand())
                        {
                            cmd.CommandTimeout = 600;
                            if (connection.State != ConnectionState.Open)
                                connection.Open();
                            cmd.Connection = connection;
                            cmd.CommandText = "set net_write_timeout=99999;
set net_read_timeout=99999;" + "UPDATE bigindex SET array = @array WHERE
token=@token;";
                            cmd.Prepare();
                            cmd.Parameters.AddWithValue("@token", token);
                            etArray = ds.GetXml();
                            cmd.Parameters.AddWithValue("@array", etArray);
                            cmd.ExecuteNonQuery();
                            connection.Close();
                        }
                        break;
                    }
                }

                if (docIdVarmi == false)
                {
                    DataRow row = dt.NewRow();
                    row["docId"] = docId;
                    row["frekans"] = frekans;

                    dt.Rows.Add(row);

                    dt.AcceptChanges();
                    ds.AcceptChanges();

                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.CommandTimeout = 600;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.Connection = connection;
                        cmd.CommandText = "set net_write_timeout=99999; set
net_read_timeout=99999;" + "UPDATE bigindex SET array = @array WHERE
token=@token;";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@token", token);
                        etArray = ds.GetXml();
                        cmd.Parameters.AddWithValue("@array", etArray);
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }
```

```
                    }
                    //dt yi docId ile kontrol et.
                    //docId yoksa dt den bir datarow üret ve eklemeyi yap.
                    //docId varsa frekans değerini güncelle.
                    //Database de güncelleme metodunu çalıştır.
                }
                else
                {
                    DataTable dTable = new DataTable();

                    dTable.Columns.Add("docId", typeof(int));
                    dTable.Columns.Add("frekans", typeof(int));
                    DataRow row = dTable.NewRow();
                    row["docId"] = docId;
                    row["frekans"] = frekans;
                    dTable.Rows.Add(row);

                    dTable.AcceptChanges();
                    DataSet dsx = new DataSet();
                    dsx.Tables.Add(dTable);
                    dsx.AcceptChanges();

                    using (MySqlCommand cmd = new MySqlCommand())
                    {
                        cmd.CommandTimeout = 600;
                        if (connection.State != ConnectionState.Open)
                            connection.Open();
                        cmd.Connection = connection;
                        cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "UPDATE bigindex SET array = @array WHERE token=@token;";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@token", token);
                        etArray = dsx.GetXml();
                        cmd.Parameters.AddWithValue("@array", etArray);
                        cmd.ExecuteNonQuery();
                        connection.Close();
                    }
                }
            }
            else if(array == "")
            {

                DataTable dTable = new DataTable();

                dTable.Columns.Add("docId", typeof(int));
                dTable.Columns.Add("frekans", typeof(int));
                DataRow row = dTable.NewRow();
                row["docId"] = docId;
                row["frekans"] = frekans;
```

```
                        dTable.Rows.Add(row);

                        dTable.AcceptChanges();
                        DataSet ds = new DataSet();
                        ds.Tables.Add(dTable);
                        ds.AcceptChanges();

                        using (MySqlCommand cmd = new MySqlCommand())
                        {
                           cmd.CommandTimeout = 600;
                           if (connection.State != ConnectionState.Open)
                              connection.Open();
                           cmd.Connection = connection;
                           cmd.CommandText = "set net_write_timeout=99999; set net_read_timeout=99999;" + "INSERT INTO bigindex(token,array) VALUES(@token,@array) ;";
                           cmd.Prepare();
                           if (token.Length > 44)
                           {
                              continue;
                           }
                           cmd.Parameters.AddWithValue("@token", token);
                           etArray = ds.GetXml();
                           cmd.Parameters.AddWithValue("@array", etArray);
                           cmd.ExecuteNonQuery();
                           connection.Close();
                        }

                     }

                  }

               }
               catch (Exception ex)
               {
                  connection.Close();
                  Debug.WriteLine("");
                  Debug.WriteLine("### addToken ###" + " docId: " + docId + " Hata: " + ex.Message +" Hata Yeri: " + ex.StackTrace + " ## Token : " + token + " ##Array : " + etArray);
               }
               finally
               {
                  connection.Close();
               }

            }


            //public static string Zip(string text)
```

```
            //{
            //    byte[] buffer = System.Text.Encoding.Unicode.GetBytes(text);
            //    MemoryStream ms = new MemoryStream();
            //            using (System.IO.Compression.GZipStream zip = new
System.IO.Compression.GZipStream(ms,
System.IO.Compression.CompressionMode.Compress, true))
            //    {
            //        zip.Write(buffer, 0, buffer.Length);
            //    }

            //    ms.Position = 0;
            //    MemoryStream outStream = new MemoryStream();

            //    byte[] compressed = new byte[ms.Length];
            //    ms.Read(compressed, 0, compressed.Length);

            //    byte[] gzBuffer = new byte[compressed.Length + 4];
            //            System.Buffer.BlockCopy(compressed, 0, gzBuffer, 4,
compressed.Length);
            //        System.Buffer.BlockCopy(BitConverter.GetBytes(buffer.Length), 0,
gzBuffer, 0, 4);
            //    return Convert.ToBase64String(gzBuffer);
            //}

            //public static string UnZip(string compressedText)
            //{
            //    byte[] gzBuffer = Convert.FromBase64String(compressedText);
            //    using (MemoryStream ms = new MemoryStream())
            //    {
            //        int msgLength = BitConverter.ToInt32(gzBuffer, 0);
            //        ms.Write(gzBuffer, 4, gzBuffer.Length - 4);

            //        byte[] buffer = new byte[msgLength];

            //        ms.Position = 0;
            //            using (System.IO.Compression.GZipStream zip = new
System.IO.Compression.GZipStream(ms,
System.IO.Compression.CompressionMode.Decompress))
            //        {
            //            zip.Read(buffer, 0, buffer.Length);
            //        }

            //            return System.Text.Encoding.Unicode.GetString(buffer, 0,
buffer.Length);
            //    }
            //}
        }
}
```

## EK C – PAGERANKCALCULATOR UYGULAMASI

### Program.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PageRankCalculator
{
   class Program
   {
      static void Main(string[] args)
      {
         try
         {
            string connectionString = @"server=localhost;userid=root;password=karlik;database=search_engine;SslMode=none";
            MySqlConnection con = new MySqlConnection(connectionString);
            MySqlCommand command = new MySqlCommand();
            command.Connection = con;
            MySqlDataAdapter adp = new MySqlDataAdapter();
            command.CommandText = "Select count(*) from pagecollection";
            DataTable dtCount = new DataTable();
            adp.SelectCommand = command;
            adp.Fill(dtCount);
            int total = int.Parse(dtCount.Rows[0][0].ToString());
            int baslangic = 0;
            while (baslangic != total)
            {
               MySqlConnection cn = new MySqlConnection(connectionString);
               using (MySqlCommand cmd = new MySqlCommand())
               {
                  cmd.Connection = cn;
                  MySqlDataAdapter ad = new MySqlDataAdapter();
                  cmd.CommandText = "Select * from pagecollection where counted = 0 LIMIT 1000";
                  DataTable dtx = new DataTable();
                  ad.SelectCommand = cmd;
                  ad.Fill(dtx);


                  if (dtx.Rows.Count == 0)
```

```
                    {
                        //return;
                        cn.Open();
                        using (MySqlCommand cmd3 = new MySqlCommand())
                        {
                            cmd3.Connection = cn;
                            cmd3.CommandText = "UPDATE pagecollection SET
counted = 0";
                            cmd3.CommandTimeout = 60000;
                            cmd3.ExecuteNonQuery();
                            cn.Close();
                            baslangic++;
                            continue;
                        }

                    }

                    foreach (DataRow dr in dtx.Rows)
                    {
                        try
                        {
                            using (MySqlCommand cmd2 = new MySqlCommand())
                            {
                                cmd2.Connection = cn;
                                cmd2.CommandText = "SELECT * FROM links WHERE
link = '" + dr["url"].ToString() + "'";
                                DataTable dt = new DataTable();
                                ad.SelectCommand = cmd2;
                                ad.Fill(dt);


                                //Formül 0.15 + 0.85 * (pr(p1) / c(p1) + pr(p2) / c(p2) +
...)
                                double lastPagerank = 0.15f;
                                foreach (DataRow drLink in dt.Rows)
                                {
                                    try
                                    {
                                        using (MySqlCommand cmd3 = new
MySqlCommand())
                                        {
                                            cmd3.Connection = cn;
                                            cmd3.CommandText = "Select * from
pagecollection WHERE url = '" + drLink["url"].ToString() + "'";
                                            DataTable dt2 = new DataTable();
                                            ad.SelectCommand = cmd3;

                                            ad.Fill(dt2);

                                            if (dt2.Rows.Count == 0)
```

```
                                    {
                                        continue;
                                    }

                                    double linkCount = 
Double.Parse(dt2.Rows[0]["linkCount"].ToString());
                                    double pageRank = 
Double.Parse(dt2.Rows[0]["pagerank"].ToString());


                                    if (linkCount == 0)
                                       continue;
                                    else
                                    {
                                       lastPagerank += 0.85f * (pageRank / 
linkCount);
                                    }
                                 }

                              }
                              catch (Exception ex)
                              {
                                 Debug.WriteLine(drLink["url"].ToString() + " " + 
ex.Message + ex.StackTrace);
                                 continue;
                              }

                           }

                           using (MySqlCommand cmd4 = new MySqlCommand())
                           {
                              cn.Open();
                              cmd4.Connection = cn;
                              cmd4.CommandText = "UPDATE pagecollection SET 
counted = 1, pagerank= @pagerank WHERE id = '" + dr["id"].ToString() + "'";
                              cmd4.Parameters.AddWithValue("@pagerank", 
lastPagerank);
                              cmd4.ExecuteNonQuery();
                              cn.Close();
                           }

                        }
                     }
                     catch (Exception ex)
                     {
                        Debug.WriteLine(ex.Message + " " + ex.StackTrace);
                        cn.Open();
                        using (MySqlCommand cmd4 = new MySqlCommand())
                        {
                           cmd4.Connection = cn;
```

```
                                cmd4.CommandText = "UPDATE pagecollection SET
counted = 1 WHERE id = '" + dr["id"].ToString() + "'";
                                cmd4.ExecuteNonQuery();
                                cn.Close();
                            }

                        }
                    }
                }

            }

        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex.Message + " " + ex.StackTrace);

        }
    }

}
}
```

**EK D - WEBSEARCHER UYGULAMASI**

**Default.aspx.cs**

```
            using MySql.Data.MySqlClient;
        using System;
        using System.Collections;
        using System.Collections.Generic;
        using System.Data;
        using System.Globalization;
        using System.IO;
        using System.Linq;
        using System.Text;
        using System.Web;
        using System.Web.UI;
        using System.Web.UI.WebControls;
        using System.Xml;

        namespace WebSearcher
        {
           public partial class Default : System.Web.UI.Page
           {
              public static string connectionString =
@"server=localhost;userid=root;password=karlik;database=search_engine;SslMode=no
ne";

              protected void Page_Load(object sender, EventArgs e)
              {
              }
              DateTime dtMili = new DateTime(), dtMili2 = new DateTime();
              private void BindRepeater()
              {
                 try
                 {
                    DataSet dsCaching = new DataSet();

                    dtMili = DateTime.Now;

                    //Do your database connection stuff and get your data
                    MySqlConnection cn = new MySqlConnection(connectionString);
                    MySqlCommand cmd = new MySqlCommand();
                    cmd.Connection = cn;
                    MySqlDataAdapter ad = new MySqlDataAdapter(cmd);



                    List<string> list = new List<string>();
                    list = txtSearch.Text.Split(' ').ToList();
                    for (int i = 0; i < list.Count; i++)
```

```
                {
                    list[i] = list[i].Trim();
                }
                list = list.Distinct().ToList();

                list = list.ConvertAll(d => d.ToLower());
                list.RemoveAll(item => item.ToLower() == "in");
list.RemoveAll(item => item.ToLower() == "a"); list.RemoveAll(item =>
item.ToLower() == "or"); list.RemoveAll(item => item.ToLower() == "and");
                list.RemoveAll(item => item.ToLower() == "b"); list.RemoveAll(item
=> item.ToLower() == "c"); list.RemoveAll(item => item.ToLower() == "c");
list.RemoveAll(item => item.ToLower() == "e");
                list.RemoveAll(item => item.ToLower() == "f"); list.RemoveAll(item
=> item.ToLower() == "g"); list.RemoveAll(item => item.ToLower() == "h");
list.RemoveAll(item => item.ToLower() == "ı");
                list.RemoveAll(item => item.ToLower() == "i"); list.RemoveAll(item
=> item.ToLower() == "j"); list.RemoveAll(item => item.ToLower() == "k");
list.RemoveAll(item => item.ToLower() == "l");
                list.RemoveAll(item => item.ToLower() == "m");
list.RemoveAll(item => item.ToLower() == "n"); list.RemoveAll(item =>
item.ToLower() == "o"); list.RemoveAll(item => item.ToLower() == "ö");
                list.RemoveAll(item => item.ToLower() == "p"); list.RemoveAll(item
=> item.ToLower() == "s"); list.RemoveAll(item => item.ToLower() == "ü");
list.RemoveAll(item => item.ToLower() == "y");
                list.RemoveAll(item => item.ToLower() == "q"); list.RemoveAll(item
=> item.ToLower() == "t"); list.RemoveAll(item => item.ToLower() == "v");
list.RemoveAll(item => item.ToLower() == "z");
                list.RemoveAll(item => item.ToLower() == "r"); list.RemoveAll(item
=> item.ToLower() == "u"); list.RemoveAll(item => item.ToLower() == "x");
list.RemoveAll(item => item.ToLower() == "w");
                list.RemoveAll(item => item.ToLower() == "veya");
list.RemoveAll(item => item.ToLower() == "ve"); list.RemoveAll(item =>
item.ToLower() == "");

                cmd.CommandText = "Select * from caching where query='";
                string query = "";
                for (int i = 0; i < list.Count; i++)
                {
                    query += list[i];
                    if (list.Count != i + 1)
                    {
                        query += ",";
                    }
                    else
                    {
                        cmd.CommandText += query + "'";
                    }
                }

                //save the result in data table
```

```
                ad.SelectCommand = cmd;
                ad.Fill(dsCaching);

                DataSet ds;

                if (dsCaching.Tables[0].Rows.Count < 1)
                {
                   DataTable sonuc = new DataTable();

                   sonuc.Columns.Add("url", typeof(string));
                   sonuc.Columns.Add("title", typeof(string));
                   sonuc.Columns.Add("description", typeof(string));
                   sonuc.Columns.Add("rank", typeof(double));
                   sonuc.Columns.Add("drawed", typeof(string));

                   for (int j = 0; j < list.Count; j++)
                   {
                      cmd.CommandText = "Select * from bigindex where token='" +
list[j].ToLower() + "'";

                      //save the result in data table
                      DataTable dt = new DataTable();
                      ad.SelectCommand = cmd;
                      ad.Fill(dt);

                      ds = new DataSet();


                      try
                      {
                         ds.ReadXml(new
StringReader(dt.Rows[0]["array"].ToString()));
                      }
                      catch (Exception ex)
                      {
                         continue;
                      }


                      dt = ds.Tables[0];

                      int index = 0, len = 0;
                      string urlD = "";
                      try
                      {
                         for (int i = 0; i < dt.Rows.Count; i++)
                         {
                            DataRow dr = sonuc.NewRow();
```

```
                    cn = new MySqlConnection(connectionString);
                    cmd = new MySqlCommand();
                    cmd.Connection = cn;
                    ad = new MySqlDataAdapter(cmd);
                    cmd.CommandText = "Select * from pageCollection where
id='" + dt.Rows[i]["docId"].ToString() + "'";

                    //save the result in data table
                    DataTable dtx = new DataTable();
                    ad.SelectCommand = cmd;
                    ad.Fill(dtx);

                    try
                    {
                       urlD = dtx.Rows[0]["url"].ToString();
                    }
                    catch (Exception)
                    {
                       continue;
                    }

                    double                pagerank                =
(Double.Parse(dtx.Rows[0]["pagerank"].ToString()) * 1000);
                    double                 frekans                =
Double.Parse(dt.Rows[i]["frekans"].ToString());
                    dr["rank"] = pagerank + frekans;

                    string icerik = dtx.Rows[0]["icerik"].ToString();
                    string icerikLower = icerik.ToLower();
                    var          unaccentedIcerik          =          String.Join("",
icerikLower.Normalize(NormalizationForm.FormD).Where(c                =>
char.GetUnicodeCategory(c) != UnicodeCategory.NonSpacingMark)).Replace("ı", "i");


                    for (int y = 0; y < list.Count; y++)
                    {
                       var          unaccentedListEleman          =          String.Join("",
list[y].Normalize(NormalizationForm.FormD).Where(c                =>
char.GetUnicodeCategory(c) != UnicodeCategory.NonSpacingMark)).Replace("ı", "i");

                       index          =          unaccentedIcerik.IndexOf("          "          +
unaccentedListEleman.ToLower());
                       int son1index = index;
                       int sonindex = index;
                       if (index < 0)
                          continue;

                       bool don = true;

                       while (don)
```

```
                {
                    sonindex--;
                    if (sonindex <= 0)
                    {
                        don = false;
                        continue;
                    }

                    string substr = unaccentedIcerik.Substring(sonindex, 2);

                    if (substr == ". ")
                    {
                        if (sonindex >= index - 90)
                            index = sonindex;
                        don = false;
                    }

                }

                don = true;
                index = son1index;
                sonindex = son1index;
                if (index == son1index)
                {
                    index = --sonindex;
                    while (don)
                    {
                        sonindex--;
                        if (sonindex < 0)
                        {
                            don = false;
                            index = 0;
                            continue;
                        }

                        string substr = unaccentedIcerik.Substring(sonindex,
1);

                        if (substr == " ")
                        {
                            if (sonindex >= index - 90)
                                index = sonindex;
                            don = false;
                        }

                    }
                }

                if (index <0)
                {
```

```
                    index = son1index;
                }

                if (index + 100 < icerik.Length)
                {
                    dr["description"] = icerik.Substring(index + 1, 100) +
"...";
                    break;
                }
                else
                {
                    dr["description"] = icerik.Substring(index + 1,
icerik.Length - index - 10) + "...";
                    break;
                }
            }


            dr["url"] = dtx.Rows[0]["url"].ToString();

            for (int y = 0; y < list.Count; y++)
            {
                string url = dr["url"].ToString();
                var unaccentedListEleman = String.Join("",
list[y].Normalize(NormalizationForm.FormD).Where(c =>
char.GetUnicodeCategory(c) != UnicodeCategory.NonSpacingMark)).Replace("ı", "i");
                double rank = Double.Parse(dr["rank"].ToString());
                if (url.Contains(unaccentedListEleman))
                {
                    dr["rank"] = rank + 100;
                }

                string title = dtx.Rows[0]["title"].ToString();
                string titleLower = title.ToLower();
                var unaccentedTitle = String.Join("",
titleLower.Normalize(NormalizationForm.FormD).Where(c =>
char.GetUnicodeCategory(c) != UnicodeCategory.NonSpacingMark)).Replace("ı", "i");

                rank = Double.Parse(dr["rank"].ToString());
                if (unaccentedTitle.Contains(unaccentedListEleman))
                {
                    dr["rank"] = rank + 100;
                }



                if (title == "")
                {
                    title = icerik.Substring(0, 100) + "...";
                }
```

```
                        dr["title"] = title;


                        if (!unaccentedIcerik.Contains(unaccentedListEleman))
                        {
                            dr["drawed"] = dr["drawed"].ToString() + " " + list[y];
                        }

                    }

                    dr["drawed"] = dr["drawed"].ToString().Split('
').Count().ToString() + " " + dr["drawed"];



                    sonuc.Rows.Add(dr);
                }

            }
            catch (Exception ex)
            {
                Console.Write(ex.Message + " " + urlD + " " + ex.StackTrace);
            }
        }



        for (int y = 0; y < sonuc.Rows.Count; y++)
        {
            string URL = sonuc.Rows[y]["url"].ToString();
            for (int e = (y + 1); e < sonuc.Rows.Count; e++)
            {
                if (URL == sonuc.Rows[e]["url"].ToString())
                {
                    if (Double.Parse(sonuc.Rows[y]["rank"].ToString()) >=
Double.Parse(sonuc.Rows[e]["rank"].ToString()))
                    {
                        sonuc.Rows.Remove(sonuc.Rows[e]);
                        e--;
                    }
                    else
                    {
                        sonuc.Rows.Remove(sonuc.Rows[y]);
                        y--;
                        break;
                    }
                }
            }
        }
```

```
                for (int y = 0; y < sonuc.Rows.Count; y++)
                {

                    for (int i = 0; i < list.Count; i++)
                    {


                        string description = sonuc.Rows[y]["description"].ToString();
                        var unaccentedDescription = String.Join("",
description.ToLower().Normalize(NormalizationForm.FormD).Where(c =>
char.GetUnicodeCategory(c) != UnicodeCategory.NonSpacingMark)).Replace("ı", "i");

                        var unaccentedListEleman = String.Join("",
list[i].Normalize(NormalizationForm.FormD).Where(c => char.GetUnicodeCategory(c)
!= UnicodeCategory.NonSpacingMark)).Replace("ı", "i");

                        int index =
unaccentedDescription.IndexOf(unaccentedListEleman);
                        if (index != -1)
                        {
                            var metin = description.Substring(index, list[i].Length);
                            description = description.Replace(metin, "<b>" + metin +
"</b>");
                        }
                        sonuc.Rows[y]["description"] = description;


                        for (int j = 0; j < list.Count; j++)
                        {
                            unaccentedListEleman = String.Join("",
list[j].Normalize(NormalizationForm.FormD).Where(c => char.GetUnicodeCategory(c)
!= UnicodeCategory.NonSpacingMark)).Replace("ı", "i");

                            int ikinciindex =
unaccentedDescription.IndexOf(unaccentedListEleman);

                            if (index != -1 && ikinciindex != -1 && index !=
ikinciindex && index - ikinciindex > -20 && index - ikinciindex < 20)
                            {
                                double rank =
Double.Parse(sonuc.Rows[y]["rank"].ToString());
                                sonuc.Rows[y]["rank"] = rank + 100;
                            }
                        }

                    }

                }

                DataView dvSonuc = sonuc.DefaultView;
```

```
                    dvSonuc.Sort = "drawed asc,rank desc";
                    for (int y = 0; y < sonuc.Rows.Count; y++)
                    {

                       sonuc = dvSonuc.ToTable();
                       for (int i = 0; i < sonuc.Rows.Count; i++)
                       {

                          int length = sonuc.Rows[i]["drawed"].ToString().Length;
                          int index = sonuc.Rows[i]["drawed"].ToString().IndexOf(' ');
                          string str = sonuc.Rows[i]["drawed"].ToString();

                          if (str.Length > 1)
                             sonuc.Rows[i]["drawed"] = str.Substring(index + 1, length -
index - 1);
                          else
                             sonuc.Rows[i]["drawed"] = "";

                          if (sonuc.Rows[i]["title"].ToString() == "")
                             sonuc.Rows[i]["title"]                                    =
sonuc.Rows[i]["description"].ToString();

                       }


                    }
                    Session["sonuc"] = sonuc;

                    ds = new DataSet();
                    ds.Tables.Add(sonuc);
                    ds.AcceptChanges();
                    MySqlCommand cmd2 = new MySqlCommand();
                    cmd2.Connection = cn;
                    cn.Open();
                    cmd2.CommandText = "INSERT INTO caching(query,array)
VALUES(@query,@array) ;";
                    cmd2.Prepare();

                    cmd2.Parameters.AddWithValue("@query", query);
                    cmd2.Parameters.AddWithValue("@array", ds.GetXml());
                    cmd2.ExecuteNonQuery();
                    cn.Close();

                    // caching tablosuna ekleme yap.


                 }
                 else
                 {
                    //Caching tablosundan çek göster.
```

```
                    DataSet sonuc = new DataSet();
                    string dataString =
dsCaching.Tables[0].Rows[0]["array"].ToString();
                    StringReader sr = new StringReader(dataString);
                    sonuc.ReadXml(sr);

                    Session["sonuc"] = sonuc.Tables[0];
                }
                dtMili2 = DateTime.Now;
                geldi = true;
                Pager();

            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);

                rptPaging.Visible = false;
                RepCourse.Visible = false;
                pnlSonucYok.Visible = true;
                lblTime.Text = "";
            }



        }
        bool geldi = false;
        double ms;
        private void Pager()
        {

            DataTable sonuc = (DataTable)Session["sonuc"];
            if (sonuc.Rows.Count > 0)
            {
                if (geldi)
                {
                    TimeSpan span = dtMili2 - dtMili;
                    ms = (double)span.TotalSeconds;
                    geldi = false;
                }

                lblTime.Text = "Toplamda " + sonuc.Rows.Count.ToString() + "
sonuç bulundu (" + ms.ToString() + " saniye)";
                rptPaging.Visible = true;
                RepCourse.Visible = true;
                pnlSonucYok.Visible = false;
                PagedDataSource pgitems = new PagedDataSource();
                pgitems.DataSource = sonuc.DefaultView;
                pgitems.AllowPaging = true;
```

```
                //Control page size from here
                pgitems.PageSize = 7;
                pgitems.CurrentPageIndex = PageNumber;
                if (pgitems.PageCount > 1)
                {
                    rptPaging.Visible = true;
                    ArrayList pages = new ArrayList();
                    for (int i = 0; i <= pgitems.PageCount - 1; i++)
                    {
                        pages.Add((i + 1).ToString());
                    }
                    rptPaging.DataSource = pages;
                    rptPaging.DataBind();
                }
                else
                {
                    rptPaging.Visible = false;
                }

                //Finally, set the datasource of the repeater
                RepCourse.DataSource = pgitems;
                RepCourse.DataBind();
            }
            else
            {
                lblTime.Text = "";
                rptPaging.Visible = false;
                RepCourse.Visible = false;
                pnlSonucYok.Visible = true;
            }

        }
        //This method will fire when clicking on the page no link from the pager repeater
        protected void rptPaging_ItemCommand(object source, System.Web.UI.WebControls.RepeaterCommandEventArgs e)
        {
            PageNumber = Convert.ToInt32(e.CommandArgument) - 1;
            Pager();
        }
        public int PageNumber
        {
            get
            {
                if (ViewState["PageNumber"] != null)
                {
                    return Convert.ToInt32(ViewState["PageNumber"]);
                }
                else
                {
```

```
                return 0;
            }
        }
        set { ViewState["PageNumber"] = value; }
    }
    protected void btnSearch_Click(object sender, EventArgs e)
    {
        BindRepeater();
    }
}
```

**Default.aspx**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebSearcher.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Arama Kelimesi:  
            <asp:TextBox runat="server" ID="txtSearch"></asp:TextBox>
            <asp:Button runat="server" ID="btnSearch" OnClick="btnSearch_Click" Text="Arama Yap"/>
            <div style="height: 13px">

            </div>
            <asp:Panel runat="server" ID="pnlSonucYok" Visible="false">
                <br />
                <span style="margin-left:100px"></span><asp:Label runat="server" ForeColor="Red" Text="Sonuç Bulunamadı"></asp:Label>
            </asp:Panel>

            <asp:Label style="height: 20px" ID="lblTime" runat="server" ForeColor="GrayText" Text=""></asp:Label>
            <br />
            <br />
            <asp:Repeater ID="RepCourse" runat="server">

                <ItemTemplate>

                    <div style="width: 500px; font-size:13px;">
                        <a href="<%#Eval("url") %>">
                            <%#Eval("title") %>
```

```
                        </a>
                        <div style="color:chocolate">
                            <%#Eval("url") %>
                        </div>
                        <div id="description">
                            <%#Eval("description") %>
                        </div>
                        <div  style="font-size:13px;  color:gray;  text-decoration:line-
through;">
                            <%#Eval("drawed") %>
                        </div>
                        <div style="height: 11px"></div>
                    </div>

                </ItemTemplate>

            </asp:Repeater>

            <br />
            <div style="overflow: hidden;">

                <asp:Repeater              ID="rptPaging"              runat="server"
OnItemCommand="rptPaging_ItemCommand">
                    <ItemTemplate>
                        <asp:LinkButton ID="btnPage"
                            Style="padding:  8px;  margin:  2px;  background:  #ffa100;
border: solid 1px #666; font: 8pt tahoma;"
                            CommandName="Page"          CommandArgument="<%#
Container.DataItem %>"
                            runat="server" ForeColor="White" Font-Bold="True">
        <%# Container.DataItem %>
                        </asp:LinkButton>
                    </ItemTemplate>
                </asp:Repeater>

            </div>
        </div>

    </form>
</body>
</html>
```

# ÖZGEÇMİŞ

## KİŞİSEL BİLGİLER


**Adı Soyadı** : Mehmet Karlık
**Uyruğu** : T.C.
**Doğum Yeri ve Tarihi** : İzmir , 18.11.1986
**Telefon** : 05072916691
**Faks** :
**e-mail** : mehmetkarlik21@gmail.com


## EĞİTİM

| Derece | Adı, İlçe, İl | Bitirme Yılı |
|---|---|---|
| Lise : | Yunus Kent Lisesi, Merkez, Karaman | 2004 |
| Üniversite : | KTO Karatay Üniversitesi, Karatay, Konya | 2014 |
| Yüksek Lisans : | Konya Teknik Üniversitesi, Selçuklu, Konya | 2018 |
| Doktora : | | |

## İŞ DENEYİMLERİ

| Yıl | Kurum | Görevi |
|---|---|---|
| 2014-2015 | Akınsoft | Yazılım Geliştirici |
| 2015-2016 | Gökmen Otomotiv | Kıdemli Yazılım Geliştirici |
| 2017-(devam ediyor) | Savema Markalama ve Kodlama Makinaları | Kıdemli Yazılım Geliştirici |

## UZMANLIK ALANI

C++, C#, Asp.net, Sql Server, MySql, Multithreaded Programming, Object Oriented Programming, WPF, HTML, CSS, Javascript, NodeJS, Socket Programming, Web Services, QT Framework

## YABANCI DİLLER

İngilizce

## YAYINLAR

Search Engine Architecture And A Parallel Working Search Engine Application
*II. International Scientific and Vocational Studies Congress - Engineering and Natural Sciences, Ürgüp, Nevşehir, Turkey (05-08/07/2018)*