T.C.
OKAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BİG DATA (BÜYÜK VERİ) ve GELENEKSEL VERİ SAKLAMA ve İŞLEME YÖNTEMLERİNE ETKİSİ ÜZERİNE BİR ARAŞTIRMA

DOKTORA TEZİ

CAN RAZBONYALI

HAZİRAN 2016

BİG DATA (BÜYÜK VERİ) ve GELENEKSEL VERİ SAKLAMA ve İŞLEME YÖNTEMLERİNE ETKİSİ ÜZERİNE BİR ARAŞTIRMA

DOKTORA TEZİ

CAN RAZBONYALI

tarafından

T.C.
OKAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALINA

DOKTORA

derecesi şartını sağlamak için sunulmuştur

ONAYLAYAN JÜRİ

Prof. Dr. Ahmet Faik Kaşlı
Danışman

Yrd. Doç. Dr. Birim Balcı Demirci
İzleme Komitesi Üyesi

Yrd. Doç. Dr. Erdal Güvenoğlu
İzleme Komitesi Üyesi

Doç. Dr. Turgay Tugay Bilgin
Üye

Yrd. Doç. Dr. Pınar Yıldırım
Üye

HAZİRAN 2016

ÖZET

İçinde bulunduğumuz yüzyıl, bilgi çağı olarak adlandırılmaktadır. Bilgi/veri, tüm paydaşlar açısından çok değerli bir kaynak durumuna gelmiştir. Bütün kurum ve kuruluşlar bilgi/veriyi gelişim ve karar verme sürecinde yoğun olarak kullanır olmuşlardır. Büyüyen iş potansiyeli ve gelişen ilişkiler, bu bilgi/verilerin büyük bir hızla artmasına neden olmuştur. Dolayısıyla bilgi/veriyi toplamak, saklamak ve erişmek, kurum ve kuruluşların gelişim sürecinde büyük önem kazanmıştır. Gelişen bilgisayar teknolojisi ve bilgi/veriye duyulan gereksinim, bilgi/verinin saklama ve işleme teknikleri açısından sürekli bir gelişim göstermiştir.

Bu çalışmada, giderek artan düzeyde bilgi/veriyi saklamak ve bunlara erişim tekniklerini incelemek ve yeni yaklaşımların söz konusu bu tekniklere etkisini araştırmak amaçlanmıştır. Söz konusu çalışmada öncelikle geleneksel olarak nitelendirdiğimiz veri yapıları tekniklerinden söz edilmiştir. Daha sonra yine bu sınıfa giren veri tabanı ve veri madenciliği kavramları üzerinde durulmuştur. Son bölümde ise, büyük veri ve bunun geleneksel yöntemlere olan etkisi açıklanmaya çalışılmış ve örnek bir uygulama verilmiştir.

ANAHTAR KELİMELER

Veri, bilgi, bellek, saklama, erişim, veri tabanı, veri ambarı, veri madenciliği, büyük veri.

ABSTRACT

The current century we live in is defined as the information age. Because of this, information/data has become a valuable resource for all stakeholders. All organisations are now intensely using information/data in their decision making processes. Growing business potentials and evolving relationships caused information/data to increase very rapidly. Thus, collecting, storing and accessing information/data has achieved a major importance in organisations' development processes. The innovations in computer technology and the demand for information/data have shown a continuous improvement from the aspect of storing and processing information/data.

This study aims to investigate the continuously improving information/data storage and access techniques and the effects of new approaches on these techniques. In the study, firstly traditional data structure techniques are mentioned. Then the study goes on to explain the concepts of traditional database and data mining. In the final section, Big Data and its effect on traditional methods have been explained including the application of a typical example.

KEYWORDS

Data, information, memory, storage, access, database, data warehouse, data mining, big data.

TEŞEKKÜR

Çalışmalarım süresince destek, yönlendirme ve yardımlarını esirgemeyen, her zaman dostça yaklaşarak beni motive eden danışmanım Prof. Dr. Ahmet Faik Kaşlı'ya minnettarım. Lisans öğrenimim sırasında kendilerinden ders alma şansına sahip olduğum Yrd. Doç. Dr. Birim Balcı Demirci ve Yrd. Doç. Dr. Erdal Güvenoğlu'na, Tez İzleme Komitesi'nde yer alarak verdikleri katkılardan dolayı müteşekkirim. Böyle bir çalışmaya yönelmemi sağlayan Yrd. Doç. Dr. Aydın Carus ve Doktora Program Koordinatörü Prof. Dr. B. Tevfik Akgün'e teşekkür ederim. Çalışmamın uygulama ortamının hazırlanmasına katkı veren Amazon Web Service destek ekibi ile, deneyler ve tezin düzenlenmesinde yardım ve desteklerini esirgemeyen Doç. Dr. Turgay Tugay Bilgin ve Yrd. Doç. Dr. Erdal Güvenoğlu'na müteşekkirim.

Hayatım boyunca sevgi ve desteğini üzerimden eksik etmeyen sevgili anne, baba ve kardeşime teşekkürü bir borç bilirim.

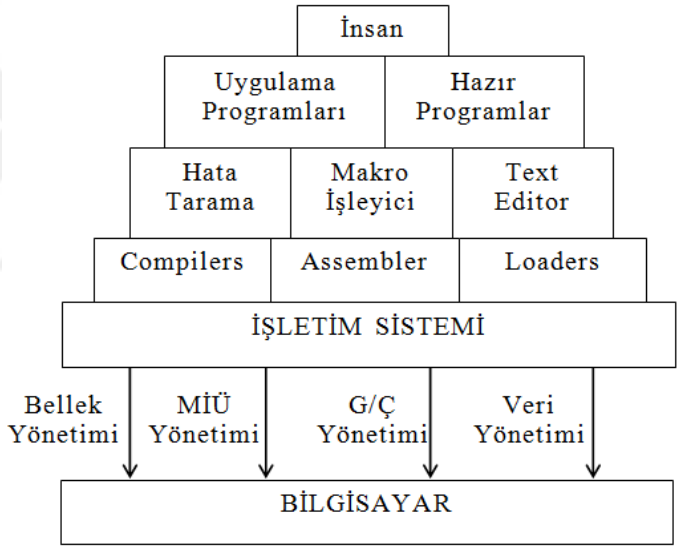
İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

1. GİRİŞ

Bir bilgisayar sisteminin genel olarak dört bileşeni vardır (Şekil 1.1). Bunlar; donanım (MİÜ, bellek, G/Ç üniteleri), işletim sistemleri, uygulama programları (compilers, assembler, loaders, database systems, vb.) ve kullanıcılar (insanlar, diğer bilgisayarlar)'dır. Temel bilgisayar kaynakları donanım tarafından sağlanmaktadır. Bu kaynakların kullanımı, kullanıcı sorunlarının çözümü için yazılmış olan uygulama programlarıyla gerçekleştirilmektedir. İşletim sistemi ise, uygulama programlarıyla donanım arasındaki iletişimi sağlamaktır (Andrew, 1987; Bartee, 1985).



Şekil 1.1. Bilgisayar Sisteminin Bileşenleri

Bilgisayar sisteminde en önemli birimlerden biri ana bellektir. Böylesine önemli bir birimin veri saklama ortamı olarak en uygun biçimde kullanılması başlıca amaç ve zorunluluktur. Bu zorunluluk bizi verilerin yapısını incelemeye ve veri yapıları ile bellek arasındaki ilişkileri araştırmaya yöneltmiştir. “En uygun biçimde kullanım” deyimi; erişim hızı, küçük bellek yeri kullanımı ve uygulanan yöntemin kolaylığı gibi özellikleri belirlemektedir. Bilgisayar belleğinin en uygun biçimde kullanılmasında verinin yanında, yapısal ilişkilerin ve bellekte veri yapısının düzenlenmesi için ortaya çıkan sorunların çözümlenmesi için,

veri yapılarına ilişkin yöntemler geliştirilmiştir. Verinin bellekte tutulma şeklini veya düzenini gösteren yöntemler, veri yapısı olarak tanımlanmaktadır. Bir veri yapısı incelenirken;

- veri yapısı tanımı,
- veri yapısı gösterimi,
- veri yapısı denetim gurubuna erişim,
- veri öğesine erişim,
- kullanım ve önemli algoritmalar,

aşamalarından geçerek belirlemek hem kuramsal açıklık sağlamakta, hem de güvenli program yazmaya yöneltmektedir (Horowitz ve Sahni, 1982).

Veri yapısı tanımı, veri yapısının kullanıcı algoritmalar tarafından görülen soyut biçimidir. Veri yapısının zorunlu başlangıç değerleri ve geçerlilik göstergeleri bu aşamada tanımlanmaktadır. Veri yapısı gösterimi, veri yapısının bilgisayar belleğinde yerleşim biçimidir. Kullanılan bilgisayarın bellek sözcük boyu da dikkate alınarak bellekte uygun bir yerleşim gerçekleştirilmektedir. Satır önce, sütun önce ve sıradüzen diziler ile yerleşim olmak üzere üç temel yerleşim düzeni söz konusudur. Veri yapısı denetim gurubuna erişim, bilgisayar belleğinin herhangi bir kesiminde yerleştirilmiş olan veri yapısı gösteriminin başlangıcına erişimdir. Bir başka deyişle veri yapısının sembolik adı ile bellekteki yerleşim adresinin eşleştirilmesidir. Veri öğesine erişim, veri yapısı denetim gurubundan geçilirken geçerliliğin sağlanmasıdır. Üst düzey programlama dillerinde derleyici kendisi için tanımlı olan veri yapısı tanımının söz dizin çözümlemesini yapınca, belirtilen parametreleri oluşturup veri yapısının denetim gurubunu kurmaktadır. Denetim gurubu kurulduktan sonra veri öğesine erişimde, veri yapısı denetim gurubunun güvenilirlik denetiminden geçmiş olan erişimin bellekte doğru adrese ulaşımı sağlanmaktadır.

Verinin belleğe etkin bir şekilde yerleştirilmesi ve yerleştirilen bu veriye erişim ve işleme için, bilgisayar teknolojinin gelişim sürecine paralel olarak çeşitli yöntemler geliştirilmiştir (Horowitz ve Sahni, 1976).

2. GELENEKSEL VERİ SAKLAMA VE İŞLEME YÖNTEMLERİ

Geleneksel veri saklama ve işleme yöntemleri denildiği zaman ilk akla gelen yöntemler; basit ve basit olmayan veri yapıları, veri tabanı ve veri madenciliğidir.

2.1. Veri Yapıları

Veri yapıları, verinin bellekte saklama ve bu saklamanın düzenini göstermektedir. Veri yapıları, basit ve basit olmayan veri yapıları şeklinde karşımıza çıkmaktadır.

2.1.1. Basit Veri Yapıları

Basit veri yapıları olarak tanımlanan bilgiler, basit rakam ve harflerdir (Çölkesen, 2004). Bir byte'da temsil edilen bu bilgiler, bilgisayar belleğinin adreslenebilen en küçük birimini oluşturmaktadır. Bu şekilde tanımlanan basit veri yapıları, sayısal ve karakter olarak sınıflandırılmaktadır. Bu sınıflamaya ek olarak mantıksal ve pointer (gösterge) veri yapıları da söz konusudur.

2.1.1.1. Sayısal bilgiler

(i). Desimal Sayılar

0 – 9 arası desimal rakamlar dört bit'le gösterilebilir. Dolayısıyla bir byte'a iki rakam konabilir. Ancak en son basamak, işaret basamağı olarak kullanılır. On altı byte'lık desimal sayılara kadar işlem yapılabilir. Bu da otuz bir rakamlık sayı demektir. Desimal sayılar, ikili kodlanmış desimal (binary coded decimal) olarak gösterilir. İşaret zone adı verilen son basamakta belirtilmektedir.

(ii). İkili Tamsayılar

Desimal sayıların uzunlukları belli sınırlar içinde değişebilir. Fakat ikili tamsayılarda durum farklıdır. Genellikle ikili tamsayıların uzunluğu otuz iki bit'lik tam kelimedir. Ancak birçok işlemde on altı bit'lik yarım kelimelik tam sayılar da kullanılabilir. Böylece bellek daha etkin kullanılmış olur.

İkili tam sayılarda en soldaki bit, sayının işaretini belirler. Bu bit sıfır ise sayı pozitif, bir ise sayı negatiftir. Bu işaret bit'i ister sıfır ister bir olsun, sayının değeri hesaplanırken dikkate alınmaz. Pozitif sayılar işaret bit'i sıfır şeklinde ikili olarak belirtilir. Negatif sayılar ise, işaret bit'i bir şeklinde ve ikinin tümleyicisi olarak belirtilir. İkinin tümleyicisini elde etmek için, her bit'in tersi alınmakta ve sağdaki kısma bir eklenmektedir.

(iii). Kesirli Sayılar

Kesirli sayılar bilgisayarda kayan noktalı (floating point) ya da sabit noktalı (fixed point) olarak tanımlanmaktadır. Onun üssü şeklinde ifade edilen bu tür sayılarda, onun üssü hangi yönde ve ne kadar kayılacağını göstermektedir. Artı üs, sağa doğru üssün değeri kadar kayılacağını, yani sayının on'un üssü ile çarpılacağını göstermektedir. Üssün işaretinin sayının işareti üzerinde hiçbir etkisi yoktur. Bu yapısal özellik nedeniyle, yarım kelime kayan noktalı sayı kullanımı yoktur. Kısa kayan noktalı sayı otuz iki bit, uzun kayan noktalı sayı ise altmış dört bit uzunluğundadır. Kayan noktalı sayının işaretini, ikili tam sayılarda olduğu gibi en sol bit gösterir. Bu bit sıfır ise pozitif bir sayı, bir ise negatif bir sayı anlamındadır. Kısa ve uzun kayan noktalı sayıların her ikisinde de işaret bitinden sonra gelen yedi bit'e karakteristik denilmektedir. Logaritmalarda olduğu gibi, karakteristik noktanın kayış yönünü ve miktarını belirler. Kayış yönü, bu yedi bitin değerine göre saptanmaktadır. Karakteristik ile ifade edilebilecek rakam 127'dir. Bu rakamın yarısı olan 64'den büyük

sayılar sağa, küçük olanlar sola kaymayı, 64 ise üssün sıfır olduğunu göstermektedir. Kayma miktarı ise yedi bitin tamamından 64 çıkartılarak elde edilmektedir. Karakteristik dışında kalan bitlere, daima birden küçük olması nedeniyle kesir denilmektedir. Uzun veya kısa kayan noktalı işlemlerde en iyi hassasiyet, kesir kısımlar normalize ise sağlanır. Kesir kısmın normalize olması demek, noktadan sonra gelen ilk basamağın sıfır olmaması demektir.

2.1.1.2. Karakter Bilgiler

Her karakter bir byte'da gösterilebilir. Byte'da sekiz bit olduğundan, 256 farklı karakterin gösterimi mümkündür. Bilgisayar sistemlerinde karakter bilgiler ASC II ya da EBCDIC formatta gösterilir. Bu kodlama sistemleri byte'ı zone ve nümerik olmak üzere iki bölüme ayırır. Her karakter bir byte'ta ASC II kodlamada yedi bit, EBCDIC kodlamada sekiz bit ile temsil edilmektedir.

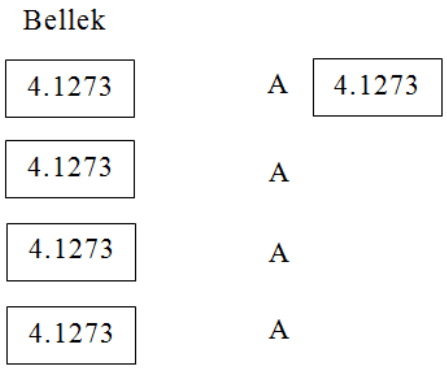
2.1.1.3. Mantıksal Bilgiler

Mantıksal basit veri yapıları TRUE (doğru) ya da FALSE (yanlış) değerini alabilen verilerdir. Kullanımları programlama dillerine göre çeşitlilik göstermekte olup, sekiz tane bir ya da sıfır ile tanımlanmaktadır. Bu yapı, erişimde hızlılık sağlamasına karşın fazla bellek harcamaktadır. Çünkü genellikle makine komutları kelime (word) üzerinden işlem yapmaktadır.

2.1.1.4. Pointer (Gösterge) Bilgisi ve Bellek Yapıları

Pointer, bir veri yapısının yerini gösteren bir gösterici ya da bağ olarak tanımlanmaktadır (Şekil 2.1). Pointer'in en belirgin özelliği, sabit uzunlukta bir veri birimi olmasıdır. Yani basit veri yapısının karmaşık olup olmamasına ya da türüne bakmaksızın bünyesinde sadece tek bir sayı

saklamasıdır. Bu da istenen verinin yerini gösteren adrestir. Pointer'ların diğer bir özelliği ise, güncelleme ve silme işlemlerini çok hızlı bir şekilde yapabilmesidir.



Şekil 2.1. Pointer Yapısı

Pointer'ların veri yapılarının yerini belirleyen bir adres olduğu belirtilmişti. Bu bilginin bellekteki konumu ise, genellikle bir kelime (word) ya da bir yarım kelime (half word)'lik sahalarda saklanmaktadır. Dolayısıyla bilgilere ilişkin adresler büyüdükçe, pointer'lar için kullanılan sahalar da büyümektedir. Bunun için pointer'daki adres, herhangi bir veri yapısını teknolojiye bağlı olarak mutlak (absolute) ve göreceli (relative) adres olmak üzere iki farklı mod'da gösterebilmektedir. Dolayısıyla pointer kullanımında bellek kullanım ve erişim etkinliği sağlanmış olur.

2.1.2. Basit Olmayan Veri Yapıları

2.1.2.1. Doğrusal Listeler (Diziler)

Doğrusal listeler; elemanları arasındaki bağlantının birbiri peşi sıra gelmeleriyle sağlanan, bellekte yan yana dizilmiş elemanlardan oluşan tek boyutlu bir dizi görünümündedir. Diğer bir deyişle doğrusal bir listedeki elemanların adresleri birbiri peşi sıra bulunmaktadır. Adresleri peşi sıra olmayan doğrusal listeler de vardır ki, bunlara bağlaçlı doğrusal listeler denilmektedir (Tremblay ve Sorenson, 1984). Doğrusal listeler bellek

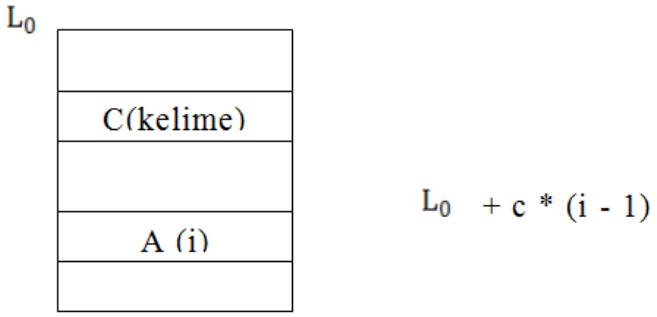
konumlarına göre, sıradan ve bağlaçlı olan bellek konumları olmak üzere ikiye ayrılmaktadır.

(1) Sıradan Olan Bellek Konumları

Bu listeler, listelerdeki bilgilerin üzerinde yapılan işlemlerin biçimine göre (ekleme, çıkarma, erişim, vb.) array (dizin), stack (yığın) ve queue (kuyruk) olmak üzere üçe ayrılmaktadır.

(i). Dizin

Dizinler, bir vektör şeklinde bilgisayar belleğinde eşit uzunluktaki aynı tipten birbiri ardı sıra gelen bilgi sahalarıdır (Şekil 2.2). Herhangi bir basit değişkenin kullanacağı değer için bellekte belirlenmiş bir yer gerekiyorsa, dizinlerde de aynı şekilde tek bir değişken adı altında eşit uzunluktaki sahalardan oluşan bir bellek gerekmektedir. Örneğin n elemanı olan bir dizi için, n tane eşit uzunluktaki bellek sahası yan yana gelecektir.



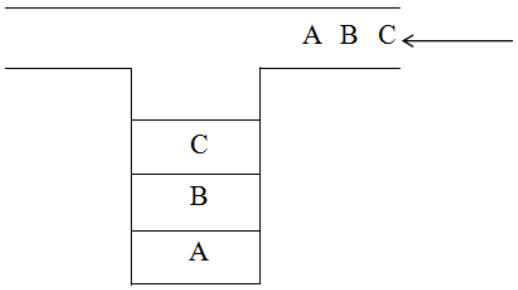
Şekil 2.2. Dizinin Bellek Görünümü

Dizinler için bellekte yer ayrılma, programlama dillerinin yapısına bağlı olarak derleme ya da uygulama aşamasında gerçekleştirilmektedir. Dolayısıyla kullanılan değişkenin statik ya da dinamik olmasına göre değişiklik gözlenir. Örneğin Fortran programlama dilinde bellekte yer ayırma derleme aşamasında, Algol ve PL/1 programlama dillerinde ise uygulama aşamasında olmaktadır. İki boyutlu olarak belirlenmiş bir

dizinin bellekteki yerleşimi, yine programlama dillerinin yapısına göre değişmektedir. Örneğin Fortran programlama dilinde sütun sütun; Algol, Pascal ve PL/1 programlama dillerinde ise iki boyutlu dizinler belleğe satır satır yerleşmektedir.

(ii). Yığın

Bu tür doğrusal listelerde, listenin yalnızca bir tarafı aktif durumdadır. Bütün işlemler listenin aktif tarafından başlanarak yapılmaktadır. Bu tür doğrusal listeler, son gelen ilk çıkar (last in first out = LIFO) deyimiyle tanımlanır (Şekil 2.3).

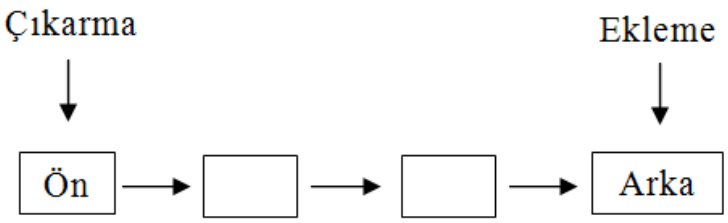


Şekil 2.3. Yığın Görünümü

Yığınlarda ekleme işlemi PUSH, çıkarma işlemi POP operatörleri ile gerçekleştirilmekte olup, bu tür uygulamalar genellikle blok yapılı programlama dillerinde değişken tablolarının düzenlenmesinde kullanılmaktadır.

(iii). Kuyruk

Bir elemana erişim ve elemanı çıkarma işlemlerinin bir uçtan, eklemelerin ise diğer uçtan yapıldığı doğrusal listelerdir. Kuyruklar, ilk gelen ilk çıkar (first in first out = FİFO) deyimiyle de tanımlanmaktadırlar (Şekil 2.4).



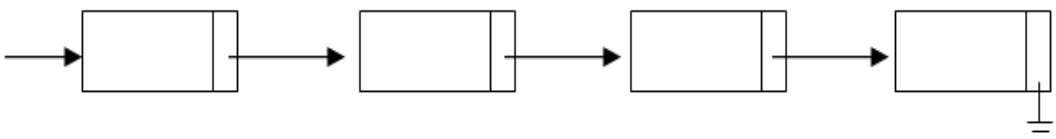
Şekil 2.4. Kuyruk Görünümü

Kuyruklar, bilgisayar sisteminde birçok işin işlem mantığını oluşturmaktadır. Örneğin işletim sistemleri bünyesindeki birçok denetim mekanizması kuyrukları kullanmaktadır. Özellikle çoklu programlama ortamında, bir anda bilgisayara çalışma kapasitesinin üstünde program girilerek kuyrukta sıraya sokulur ve sırası geldikçe çalışma ortamına alınırlar. Yine bu programların çıktıları aynı anda yazıcıdan alınamazlar. Bunun için önce bir kuyruğa alınırlar ve sırayla buradan yazıcıya gönderilirler. Bir kuyruktaki bilgilerin oluşturulması ve silinmesi ön ve arka pointerlar aracılığıyla izlenmekte ve uygulanmaktadır.

Kuyruk tipi veri yapılarında bellek kullanımını etkinleştirmek için, dairesel ve çift yönlü kuyruklar da geliştirilmiştir.

(2) Bağlaçlı Olan Bellek Konumları

Bilgiler bir dizide saklandığında, bellek kullanımı ve yeni bilgi ekleme/çıkarma konularında sorunlar doğmaktadır. Bu sorunları gidermek amacıyla, saklanması gereken bilgilerin yanında onların sıralarını gösteren bilgilerin de saklanmasıyla bağlaçlı listeler adı verilen veri yapıları geliştirilmiştir (Şekil 2.5).



Şekil 2.5. Bağlaçlı Liste

Bağlaçlı liste kullanımında daha fazla bellek harcandığı sanılırsa da, kullanılan bir hücre yeniden belleğe iade edildiği için sıradan bir listeye oranla daha az bellek yeri gerektirmektedir (Kurnaz, 2004). Bağlaçlı listelerde, ortadan bir elemanı ekleme/çıkarma işlemi daha kolay gerçekleştirilmektedir. Rastgele bir elemana erişim sıralı atamada daha kolay olmasına karşın, bağlaçlı atamada erişilmek istenen elemana kadar olan elemanların tümü taranacağından aranan elemanın ilk elemandan uzaklığına göre erişim zamanı değişmektedir. Bağlaçlı listeler sıradan işlemler için uygun olup, iki ya da daha fazla listeyi birleştirme işlemi daha kolay gerçekleştirilmektedir. Ayrıca, karmaşık yapıların temsiline olanak verdiğinden bağlaçlı listeler çok kullanışlıdır.

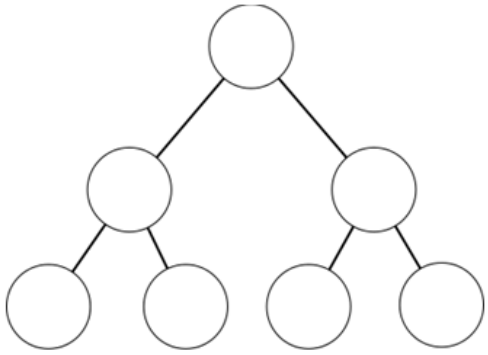
Belirtilen bu özelliklerin dışında, bağlaçlı listelerin yararlarından bir diğeri de, bilgilerin dizideki yerleri değiştirilmeden, bağ bilgileri ve liste başı bilgisi değiştirilerek bağlaçlı liste herhangi bir elemana göre sıralı hale getirilebilmektedir.

Bağlaçlı listelerin dairesel ve çift yönlü bağlaçlı listeler olmak üzere iki farklı yapısı da söz konusudur. Dairesel bağlaçlı listelerde son elemanın bağ alanında ilk elemanın adresi bulunmaktadır. Dolayısıyla listenin önü ya da arkası kavramları bu tür için geçerli değildir. Çift yönlü bağlaçlı listelerde ise, bağ bilgisi ileriye ve geriye doğru olmak üzere iki yönlüdür.

2.1.2.2. Ağaç Yapıları

Ağaç biçiminde oluşturulan ve kök, dal, yaprak, vb. kavramlara uygun olarak düzenlenen veri yapılarına ağaç yapıları denilmektedir (Şekil 2.6). Ağaç yapıları recursive olup, yukarıdaki dalların dağılım biçimi aşağıdakilerin dağılım biçiminden özde farklı değildir (Hoare, 1975). Bir ağaç, sonlu sayıda düğümden oluşan bir küme olup, kök olarak belirlenmiş bir özel düğüm ve ortak elemanı olmayan alt kümelerden oluşmaktadır. Bir

düğümün alt ağaçlarının sayısına o düğümün derecesi denir. Derecesi sıfır olan düğümler ise yaprak olarak tanımlanır. Bir ağaçta kök olarak belirlenen özel düğümün düzeyi birinci düzey alınarak, diğer düğümler bu özel düğüme (kök) göre bir düzey numarası almaktadır.



Şekil 2.6. İkili Ağaç Yapısı

Eğer bir ağaçta, alt ağaçların sırası bir birlerine göre önemli ise bunlara sıralı ağaç, önemli değil ise sırasız ağaç denilmektedir. Bilgisayar uygulamalarında sıralı ağaçlar önemli bir yer tutmakta olup, bunların içinde en önemli yeri ikili ağaçlar almaktadır. Genel olarak ağaçlar çeşitli şekillerde gösterilmekte ise de, veri yapıları yönünden düşünüldüğünde, ağaçtaki düğümler arasında mantıksal ilişkiler kurulması gerekecektir. Genellikle bu ilişkiler baba, oğul ve kardeşten oluşan aile ağaçları ile ifade edilmektedir. Ağaç türü veri yapılarının gösteriminde değişik notasyonlar kullanılmaktadır. Bunlardan bazıları dewey ondalık gösterimi, iki boyutlu bir dizinin ağaçla tanımı, söz dizim ağaçları ve ağaçların iç içe kümelerle tanımlanması olarak ifade edilebilir.

Her düğümün en fazla bir düğüme bağlandığı ya da her düğüme ait bir hücreden oluşan iki ağacın bulunduğu ağaç türlerine ikili ağaç denilmektedir. Bilgisayar belleğinde ağaç yapıları daima ikili ağaç ile temsil edilir. Çünkü ikili ağaçlarda yapılan tanımlardaki elemanların ilişkilendirilmesi için iki bağ alanı yeterlidir. Ayrıca herhangi bir ağacın ikili ağaçla tanımlanabileceği bilinmektedir.

Yaratılan bir ağaç yapısındaki elemanlara ulaşma ya da ağaçta dolaşma işleminin şekli önem kazanmaktadır. Bunun için de elemanların öncelik sırası söz konusu olabilir. Yani hangi eleman hangi elemandan önce ya da sonraki pozisyondadır gibi. Bu amaçla ağaç yapılarında dolaşma yöntemleri geliştirilmiştir. Bir ikili ağaçta aşağıda belirtilen üç temel dolaşma yöntemi vardır;

(i). Kök Önce Yöntemi (Preorder, Prefix)

Bu yönteme göre dolaşmada önce kök, sonra sol alt ağaç ve daha sonra da sağ alt ağaç işleme girmektedir.

(ii). Kök Ortada Yöntemi (Postorder, Infix)

Bu yöntemde öncelik sırası önce sol alt ağaç, sonra kök ve daha sonra da sağ alt ağaç şeklindedir.

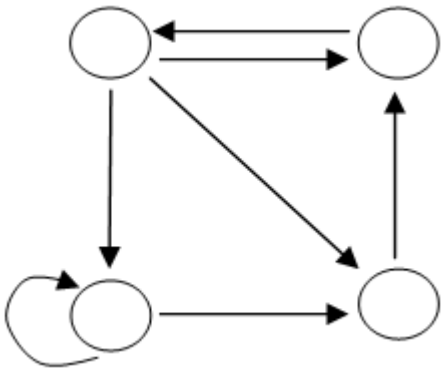
(iii). Kök Sonda Yöntemi (Endorder, Postfix)

Bu yöntemde öncelik sırası ise önce sol alt ağaç, sonra sağ alt ağaç ve daha sonra kök şeklinde gerçekleştirilmektedir.

Dolaşma yöntemleri sıralama yöntemlerinin temelini oluşturmaktadır. Sıralama yöntemlerinin uygulanmasında yığın kullanılması zorunludur. Ancak (n) hücreli bir ikili ağaçta boş bağ alanı sayısı (n+1) olduğu için, bu boş bağ alanlarının kullanımı ile yığın kullanımına gerek kalmaz. Dolayısıyla daha hızlı işlem yapma olanağı sağlanmış olmaktadır.

2.1.2.3. Graf Türü Yapılar

Aynı kümeye ait verilerin düğüm, ayrıt ve bunların birleştirilmesi ile oluşturulan yapıya graf türü veri yapısı denilmektedir (Şekil 2.7). Düğümler birleşme noktasını, ayrıtlar ise düğümler arası bağlantı ilişkisini göstermektedir (Çölkesen, 2004). Verilerin kendisi veya bir bölümü, hem düğüm hem de ayrıt bilgi bölümünde yer alabilir. Graf türü yapılarda iki yönlü ilişki söz konusu olabilmektedir. Graf türü veri yapılarında, ağaç türü yapılarda olduğu gibi sıradüzensel bir durum söz konusu değildir.



Şekil 2.7. Yönlü Graf Gösterimi

Graf türü yapılar; bilgisayar yazılımında önemli bir yer tutmakta olup, birçok sorunun çözümünde kullanılmaktadır. Örneğin, trafik ya da su taşıma altyapısının optimizasyonu graf türü yapılara yatkın uygulamalardır.

2.1.2.4. Arama (Searching) Yöntemleri

Arama yöntemleri, anahtar diye nitelendirilen bir bilgiden yararlanarak, bu bilgiye sahip olan kayıtın bulunmasını gerçekleştirmektedir (Davis, 1981). Bilgisayar belleğinde bulunan bir kayıta erişmenin amacı, o kayıtın anahtar dışında içerdiği bilgi olsa da, arama yöntemleri anahtar bilgisi dışındaki bilgileri yok saymaktadır. Çünkü aranan anahtar değerine sahip kayıta erişildiğinde, kayıtın içerdiği bilginin kullanılmasında herhangi bir güçlükle karşılaşılmaz.

Arama işlemi, çok fazla bilgisayar zamanı gerektirdiğinden uygun yöntem kullanımı bilgisayarın etkin kullanımını sağlamaktadır.

(i). Sıralı Arama (Sequential Search)

En basit arama yöntemi olarak bilinen bu yöntem, veri setindeki her kayıtın aranan anahtar değerine sahip kayıt bulununcaya kadar tek tek incelenmesi esasına dayanmaktadır. Bu inceleme sırasında aranılan anahtar değeri bulunursa arama işlemi sonlandırılır. Aranılan anahtar bilgisinin olmadığını anlamak için bütün elemanların kontrol edilmiş olması gerekmektedir. Eğer arama yapılacak bilgiler belli bir düzene konulursa, örneğin küçükten büyüğe doğru sıralanırsa, herhangi bir anahtar bilgisinin var olup olmadığını anlamak için mutlaka sona kadar arama yapılması gerekmez.

(ii). İkili Arama (Binary Search)

Sıralı bilgiler üzerinde aramada daha kısa sürede sonuç veren bir başka yöntem de ikili aramadır (Kaşlı, 2003). Bu yöntemde kontrol edilecek bilgiler sıra ile seçilmez, mevcut bilgilerin en ortasındaki olarak seçilir. Aranan bilgi ile kontrol edilen bilgi aynı ise arama başarı ile sonuçlanmış demektir. Aranan bilgi kontrol edilen bilgiden daha küçük bir değere sahip ise büyük olan bilgilerin kontrol edilmesine gerek olmayıp, arama ikiye bölünen bilgilerin küçük kısmında sürecektir. Eğer aranan bilgi kontrol edilen bilgiden daha büyük bir değere sahip ise, bu sefer arama büyük bilgiler arasında devam edecektir. Bu yönteme ikili arama isminin verilmesi, her kontrol ile mevcut kontrol edilmeyen bilgi sayısının yarıya düşürülmesidir. Bu yöntem ile eleman sayısı çok fazla olduğunda sıradan aramaya göre oldukça kısa sürede sonuca ulaşma şansı yüksektir.

(iii). Blok Arama (Block Search)

Bu yöntemde veri seti bloklara ayrılmaktadır. Her bir blok içinde kayıtların sıraya sokulması zorunlu olmayıp, blokların sırada olması yeterlidir. Eğer bloklar küçükten büyüğe doğru sıraya sokulmuş ise, birinci bloktaki kayıtların tamamının anahtar değerleri, ikinci blok içindeki kayıtların anahtar değerlerinden küçük olacaktır. Aramanın kolaylaştırılması için, her blok içindeki en büyük anahtar değerine sahip kayıtın anahtar değerinin bilinmesi yeterlidir. Veri setinin aranması, aranan kayıtın anahtar değeri ile her blok içindeki en büyük anahtar değerine sahip kayıtın anahtar değerinin karşılaştırılması şeklinde gerçekleştirilecektir. Bu işleme, aranan kayıtın anahtar değerinin bir blok içindeki en büyük anahtar değerinden küçük durum elde edilmesine kadar devam edilir. Dolayısıyla aranan kayıtı bulunduran blok saptanmış olur. Bundan sonra, o blok içinde sıradan arama yöntemi ile aranan kayıta erişim sağlanır.

(iv). Arama Ağaçları (Search Trees)

Arama işlemi veri setine bağlı olarak zaman açısından çeşitlilik göstermektedir. Dengeli ikili ağaçlar kullanılarak arama zamanı önemli ölçüde azaltılabilmektedir. Bu yöntem, oluşturulan dengeli ikili ağaçlara bilinen dolaşma yöntemleri uygulanarak gerçekleştirilmektedir.

2.1.2.5. Anahtarlama (Hashing) Yöntemleri

Veri setinde yer alan kayıtların belirli bir sırada tutulması ile arama zamanı kısaltılabilmektedir. Bununla beraber, ek olarak kayıtların sıraya sokulmasına gerek kalmaksızın daha kısa zamanda erişim olanağı sağlayan anahtardan adrese dönüşüm teknikleri de geliştirilmiştir. Bu teknikler, anahtarlama (hashing) olarak adlandırılmaktadır (Gonnet, 1984).

Çok kısa süre içinde kayıt eklenmesi, güncellenmesi ve aranması gereken veri setlerinin düzenlenmesinde anahtarlama yöntemlerinden birisinin kullanılması etkinlik sağlamaktadır. Ancak buna karşın söz konusu yöntemlerin dezavantajı, çok fazla bellek kullanmasıdır. Eğer geniş bellek kapasitesi kullanma olanağı varsa, anahtarlama yöntemi kullanılarak çok hızlı arama ve erişim gerçekleştirilebilir. Anahtarlama yöntemi, geliştirilmiş çok sayıda yöntemler topluluğudur. Bir fikir vermesi açısından; basamak analizi (digit analysis), katlama (folding), kaydırma (shifting), bölme (division), kare alma (mid-square) ve yarı şans sayısı (pseudo random) yöntemlerinden söz edilecektir.

(i). Basamak Analizi (Digit Analysis)

Bazı veri setlerinin düzenlenmesinde kayıtların yerleştirileceği adres belirlenirken, kayıtın anahtar değeri kullanılabilir. Örneğin, emekli sicil numarası veya sosyal sigortalar sicil numarası gibi. Bu yöntemin çok hızlı erişim sağlayacağı kesin olmakla birlikte, veri setinin kaplayacağı yardımcı bellek sahası anormal ölçüde büyük olacaktır
Hem bellek sahasının daha etkin kullanılabilmesi için, hem de anahtar bilgi sahasının kullanılmasıyla kayıtların uniform olarak (veri setinin her bölgesinde aynı miktarda) yerleştirilmelerini sağlamak amacıyla, sadece belirli basamakların adres olarak kullanılması tercih edilebilir. Diğer basamaklar adres oluşturmada dikkate alınmazlar. Eğer, basamak analizi yapıldıktan sonra anahtar bilgi yapısında değişiklik yapılırsa tekrar basamak analizi yapılması gerekecektir.

(ii). Katlama (Folding)

Anahtar sahasında yer alan basamak sayısının adres olarak kullanılacak bellek sahasının alabileceğinden daha fazla olduğu durumlar da söz konusudur. Bu gibi durumlarda, eğer anahtar sahasındaki basamaklar uniform (her bir basamaktaki her bir sayı tamamen şansa bağlı) olarak

dağılmışlarsa, hangi basamakların dikkate alınmayacağı konusundaki karar gelişigüzel verilebilir. Çok sayıda farklı yöntem arasından birisi, bazı basamakları gelişigüzel seçip, bu basamakların değerlerini diğer basamaklara eklemektir. Katlama olarak tanımlanan bu yöntemin amacı, adres bilgisinin elde edilmesinde anahtar bilgi sahasındaki bütün basamakların etki yapmasının sağlanmasıdır.

(iii). Kaydırma (Shifting)

Bu yöntem de katlama yöntemine benzer şekilde olup, anahtar olarak belirlenen sahanın bazı basamaklarının kaydırılarak toplanması esasına dayanmaktadır.

(iv). Bölme (Division)

İlk olarak geliştirilmiş ve uygulanması en kolay yöntemlerden biri olan bölme yöntemi, anahtar değerinin bellekte bulunabilecek adres sayısına bölünmesi şeklinde uygulanmaktadır. Bu yöntemin temeli, modüler aritmetiğe dayanmaktadır. Bir sayının (n) gibi bir bölen ile bölünmesinden (n-1) ile sıfır arasında bir artan sayı elde edilir. Yine bir sayının, bu sayının belli bir sayıya bölünmesinden arta kalan sayıya dönüştürülmesi düşüncesi, yukarıda belirtilen modüler aritmetik felsefesinden doğmuştur.

(v). Kare Alma (Mid-Square)

Bu yöntemde; anahtar değerinin kendisiyle çarpılmasından elde edilen sayı içinden, uygun sayıda basamaktan oluşan kısmının alınması ve adres olarak kullanılması önerilmektedir. Seçilen basamak sayısı, bellekte ayrılmış olan sahanın büyüklüğüne bağlı kalmaktadır. Hangi basamakların ve kaç adet basamağın kullanılacağına karar verildikten sonra, bütün kayıtların adres bilgilerinin elde edilmesinde bu basamaklar kullanılmaktadır.

(vi). Yarı Şans Sayısı (Pseudorandom)

En çok bilinen yarı şans sayısı türetme yöntemi olarak, y=(a.x+c) mod p fonksiyonu kullanılmaktadır. Bu fonksiyonda; a=1(mod 4), x için kayıt anahtar değeri, c için -1 veya 1 ve p değeri için bellek büyüklüğünü belirten ikinin üssü olan bir sayı kullanılır ve kayıt için bellek adresi olan y değeri elde edilir.

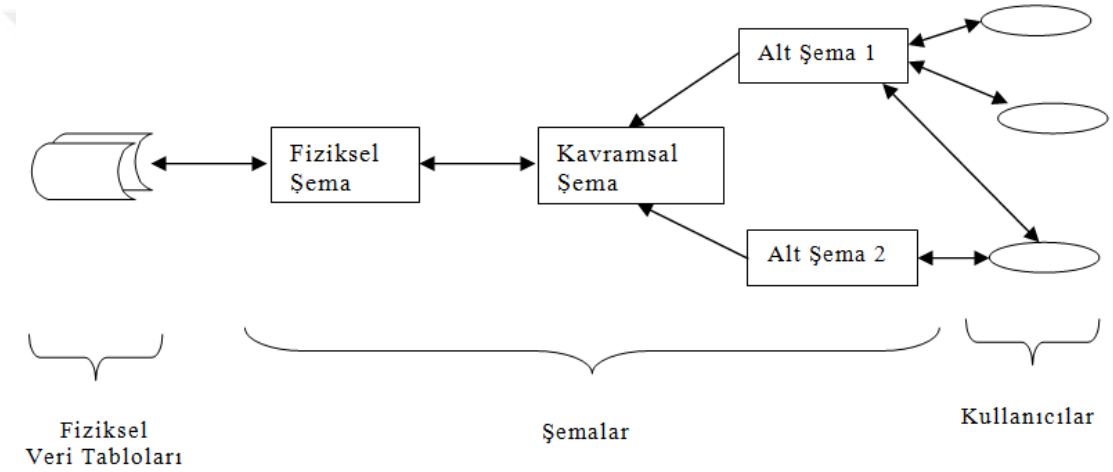
Bu yöntem, diğer anahtarlama yöntemlerinin etkisinin belirlenmesi için standart yöntem olarak da kullanılmaktadır.
İki veya daha fazla kayıtın anahtar değerleri adres değerlerine dönüştürüldüklerinde, sonuç olarak çarpışma (collision) denilen aynı adres değeri elde edilebilmektedir. Anahtarlama yönteminin seçimine karar vermeden önce, çarpışma oluştuğunda ne yapılacağına karar verilmelidir. Bu konuda karar verildikten sonra; kayıtların veri setine yerleştirilme, ekleme, çıkarma ve güncelleme işlemlerinde uygulanacak yöntemler temel olarak aynı mantığa dayanmaktadır.

Çarpışma olduğunda uygulanabilecek yöntemlerden birisi, bir sonraki adrese kayıtın eklenmesidir. Uygulanabilecek bir başka yöntem, eğer adres dolu veya kayıt aranan değil ise adres değerine belirli ve sabit bir değerin eklenmesiyle yeni adres değerinin bulunmasıdır.

2.2. Veri Tabanı

Karmaşık veri ve dosya yapıları ve çok fazla dosya arası ilişkiler ile bunlara erişim, gelinen düzeyde yetersizlik sorununu ortaya çıkarmıştır. Bu sorunu çözmek için veri saklama ve erişim konusunda yeni yazılım teknolojilerine yönelme söz konusu olmuş ve Veri Tabanı Yönetim Sistemleri (VTYS) yaklaşımı ortaya atılmıştır (Keskinel, 1985). Bu yaklaşımda veri girişi ve saklama, söz konusu veriye erişimden bağımsız

olup, kayıt ve dosya yapılarında oluşacak en küçük bir değişiklik, uygulama programlarının değişmesine ve yeniden derlenmesine neden olmaktadır. Veri tabanı sistemleri bilgisayar sistemlerinin bir bileşeni olup, birbirleriyle ilişkili veri ve programlardan oluşmaktadır. Bu veri topluluğu, veri tabanı olarak adlandırılmaktadır (Şekil 2.8). Veri tabanı bilgilerin yer aldığı ortam, veri tabanı sistemleri ise bu ortamın çeşitli yazılımlar ile yönetilmesidir. Veri tabanı, gereksinim duyulan veya duyulabilecek her türlü veriyi içermelidir (Özkan, 2009).



Şekil 2.8. Veri Tabanı Yapısı

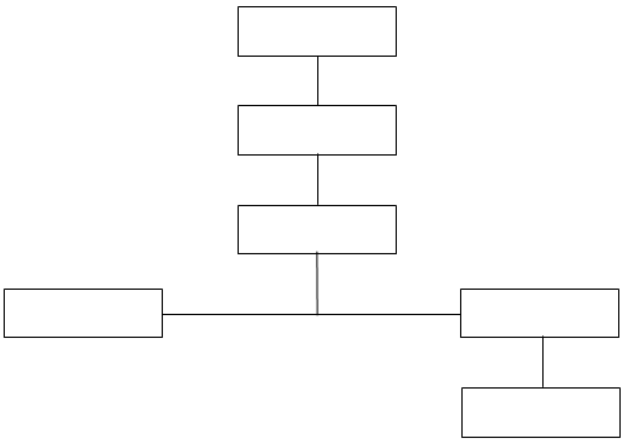
2.2.1. Veri Tabanı Yapıları

Veri tabanı yapıları; sıradüzensel (hierarchical), ağ (network) ve ilişkisel (relational) olmak üzere üç yapısal düzende karşımıza çıkmaktadır (Keskinel, 1985; Yarımağan, 2000).

(i). Sıradüzensel Veri Tabanı

Sıradüzensel veri tabanı yapısında, veri elemanları arası ilişkiler bir ağacın dalları görünümündedir (Şekil 2.9). Yani bir veri tabanı kaydı, bir kök veri elamanı ile ona bağlı alt veri elemanlarından oluşmaktadır. Kayıtlar arası ilişki birden çoğa doğrudur. Bunun anlamı her kaydın alt

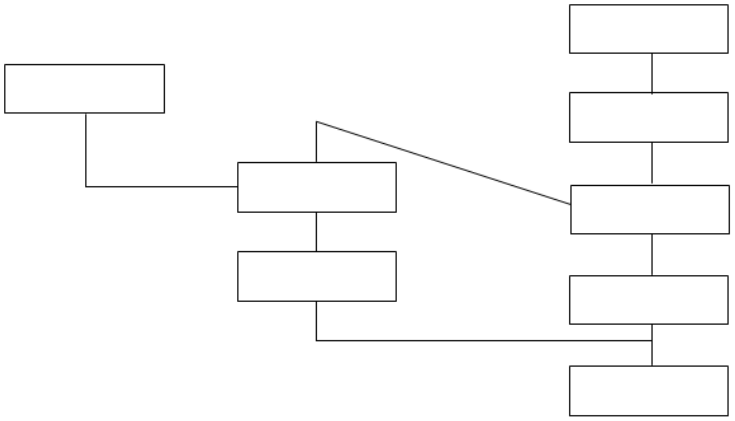
düzeyindeki birçok kayıt ile ilişki kurulabilmekte, ancak bir üst düzeyde sadece bir kayıtla ilişkilendirilebilmektedir. Kök, yapının en üst düzeyini oluşturmakta ve erişim buradan başlamaktadır. Bir sıradüzensel veri tabanı yapısından diğerine göstergeler ile erişilmektedir.



Şekil 2.9. Sıradüzensel Veri Tabanı Yapısı

(ii). Ağ Veri Tabanı

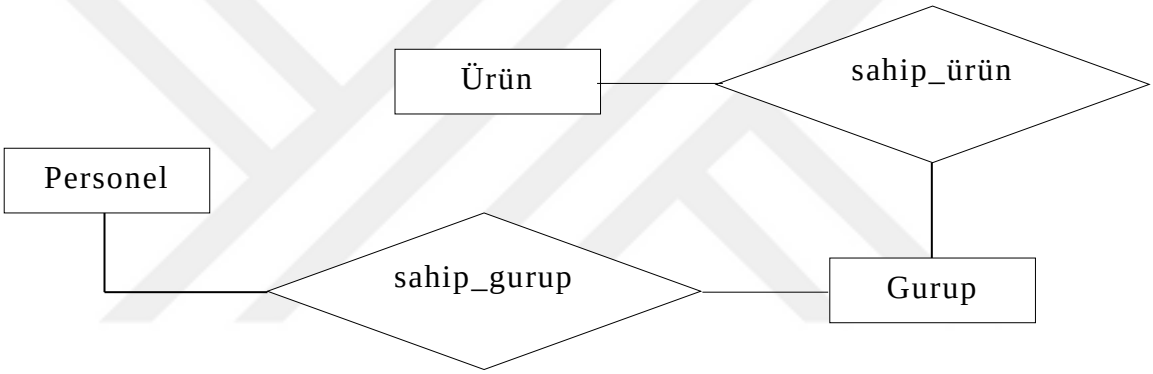
Birden fazla veri elemanıyla ilişkiye olanak veren ağ veri tabanı modeli, daha karmaşık mantıksal ilişkilerin tanımlanmasını sağlamaktadır. Bu tür yapılarda, veri tabanına değişik noktalardan erişim sağlanabilmektedir. Bunun nedeni kayıtlar arası ilişkinin çoktan çoğa olmasıdır (Şekil 2.10).



Şekil 2.10. Ağ Veri Tabanı Yapısı

(iii). İlişkisel Veri Tabanı

İlişkisel model; elemanlar arası ilişkilerin, içerdiği değerlere göre sağlanması temeline dayanmakta olup, günümüzde en yaygın biçimde kullanılan bir modeldir (Şekil 2.11). İlişkisel modelde elemanlar arası karmaşık ilişkileri basite indirgemek amaçlandığı için, tüm ilişkiler tablolar biçiminde tanımlanmaktadır. Söz konusu model, her biri özel isimlere sahip tablolardan oluşmakta ve her tablo da bir elemana veya ilişkiye karşılık gelmektedir.



Şekil 2.11. İlişkisel Veri Tabanı Yapısı

2.2.2. Veri Tabanı Yönetim Sistemi

Veri tabanı yönetim sistemi, bazı özel veri yapılarını kullanarak verilere çabuk ve kolay erişim olanağı sağlamaktadır (Martin, 1977). Bir dizi programdan oluşan veri tabanı yönetim sistemi, veri tabanlarının yaratılmasını, bakımını, kullanımını ve güvenliğini gerçekleştirmektedir.

(i). Veri Tabanının Yaratılması

Bir veri tabanının yaratılması; gerekli verilerin yapısını, kapsamını birbirleriyle olan ilişkilerini tanımlama, düzenleme ve yükleme çalışmaları olarak tanımlanmaktadır.

(ii). Veri Tabanının Bakımı

Veri eklemek, silmek, güncellemek, düzeltmek ve korumak için yapılan çalışmalar, veri tabanının bakımı olarak tanımlanmaktadır.

(iii). Veri Tabanının İşletimi

Veri tabanının işletimi, var olan veriyi kullanarak istenen bilgiyi elde etmek ve raporlama gibi işlevleri gerçekleştirmektir. Yani uç kullanıcıların ve uygulama yazılımcılarının veri tabanını kullanmasının sağlanması ve yönetimidir.

(iv). Veri Tabanının Güvenliğinin Sağlanması

Veri tabanına erişim yetkisi olanların ulaşmasına olanak veren, hatalı ve kötü kullanım sonucu zarar görmesini engelleyen ve zarar görmesi halinde onarılmasını sağlayan tüm önlemler veri tabanı güvenliği kapsamına girmektedir.

2.2.3. Veri Tabanı Yönetimi

Veri tabanı yönetimi donanım ve yazılım olgularını kapsayan yönetim ve düzenleme işlevlerini içermektedir (Keskinel, 1985). Bu işlevler, veri tabanı yöneticisi ve veri yöneticisi tarafından gerçekleştirilmektedir.

(i). Veri Tabanı Yöneticisi

Veri tabanını tasarlamak ve düzenlemek, konuya ilişkin bilgi sahibi olmayı gerektirmektedir. Bu özelliğe sahip kişiler veri tabanı yöneticisi olarak adlandırılmakta olup, veri tabanının tasarım, yaratma ve bakımını yapmaktadırlar. Ayrıca veri işletim sistemi tasarımı, donanım ve yazılım

seçimi ile günlük güncellemeler de söz konusu kişinin ilgi alanına girmektedir. Veri tabanı sisteminin performansı ve etkinliği teknik bir sorun olmakla birlikte, kullananların da sorumluluğundadır.

(ii). Veri Yöneticisi

Veri tabanı tasarımı ve yönetiminden sorumlu olan veri tabanı yöneticisinden, verilerin kopyaları ve etkinliğinden de sorumlu olması beklenmemelidir. Bu tür işlevler veri yöneticine verilmeli, veri tabanı yöneticisi de bu kişiye karşı sorumlu olmalıdır. Veri yöneticisi; veri stok, bakım ve kullanım analizlerinden birinci derece sorumlu olmalıdır. Dolayısıyla veri yöneticisinden, veri tabanı yönetici kadar teknik bilgiye sahip olması beklenmemelidir.

2.2.4. Veri Tabanı Oluşturulması

Veri tabanı oluşturulması; kullanıcı, programcı ve sistem tasarımcılarının birlikte çalışmalarıyla oluşacak bir süreçtir. Yönetim Bilişim Sistemleri açısından gereksinimler belirlenmemiş ise, veri tabanı gerçekleştirimi söz konusu olamaz. Çünkü veri/bilginin yapısal özellikleri ve birbirleri ile olan ilişkileri önemlidir. Dolayısıyla bu aşamada kullanıcının sorumluluğu önemlidir.

Öncelikle, kullanıcılar ve sistem tasarımcılar tarafından, hangi verilerin veri tabanına gireceğine ve bu veriler arasındaki mantıksal ilişkilerin neler olacağını belirleyerek veri tabanı şeması oluşturulur. Bu şemada; veri tabanında bulunacak elemanlar, veri elemanı ilişkileri ve veri tabanının yapısı gibi bilgiler yer almaktadır.

Veri tabanı şeması tasarlandıktan sonra, veri tabanına erişecek uygulamalara ilişkin alt şemalar tanımlanır. Alt şema, veri tabanı

şemasının her hangi bir uygulama programı tarafından istenen alt düzeyinin mantıksal görüntüsüdür. Her uygulama programının; veri tabanının her noktasına erişmesi gerekmediğinden, kendisiyle ilgili bölüm olan alt şemaya erişmesi yeterlidir.

Şema ve alt şema, veri tabanının mantıksal yapısına yöneliktir. Fiziksel yapının görünümü mantıksal yapıdan farklı olup, verinin fiziksel olarak yardımcı bellekte saklanma düzenini belirlemektedir. İşletim sistemi denetiminde çalışan veri yönetimi kontrol programlarının etkinliği, verinin yardımcı bellekteki fiziksel yapısıyla yakından ilişkilidir.

Oluşturulan veri tabanını kullanıcıların ulaşımı, sorgu dilleri (query language) ile gerçekleştirilmektedir. Sorgu dilleri program yazmaya gerek kalmadan, istenilen veri/bilgiye erişme olanağı sağlamaktadır.

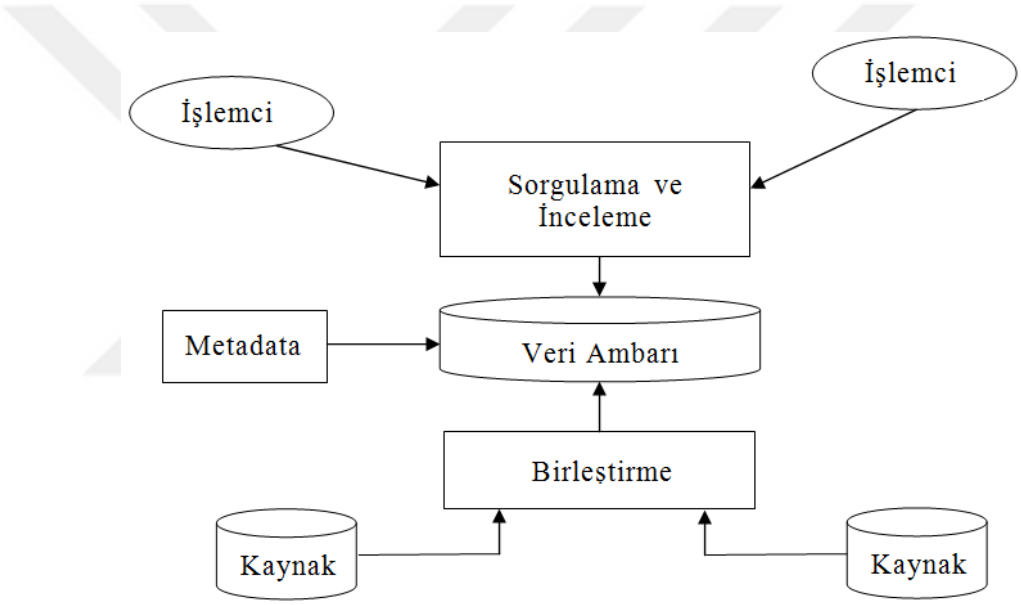
2.3. Veri Madenciliği

Gelişen teknolojik süreçte bilgisayar kullanımının artması, veri/bilgi miktarında da büyük bir artış göstermiştir. Bu artış, verilerin analizi ve veri saklama ortamlarının yeteneklerinin üzerine çıkılmasına neden olmuştur. Bu yetersizlik, veri yapıları ve veri tabanları kavramlarının dışında yeni analiz araçlarının geliştirilmesine neden olmuştur. Veri madenciliği, yoğun veri ortamında ilişki ve kuralları uygulayarak yararlı bilginin elde edilmesi olarak tanımlanmaktadır (Kaya ve Köymen, 2008).

Donanım ve yazılım teknolojilerindeki gelişim, karar destek sistemlerinin geliştirilmesinde uygun ortamların geliştirilmesini sağlamış ve veri ambarı kavramının ortaya çıkmasına neden olmuştur.

2.3.1. Veri Ambarı

Veri ambarı, karar destek sistemlerinin teknolojik alt yapısını oluşturarak var olan tüm verilerin kullanılmasını sağlamaktadır. Veri ambarı, modüler uygulamaların ilişkilendirilmesi açısından önemli olup, zaman boyutunda analitik işlemler için gereksenen bilgi alt yapısını sağlamaktadır (Şekil 2.12). Veri ambarı, karar verme sürecinde yöneticilere destek vermek amacıyla hazırlanmış, konuya yönelik, bütünleşik, zaman boyutu olan ve sadece okunabilir özelliklere sahip veri topluluğudur (Özkan, 2006).



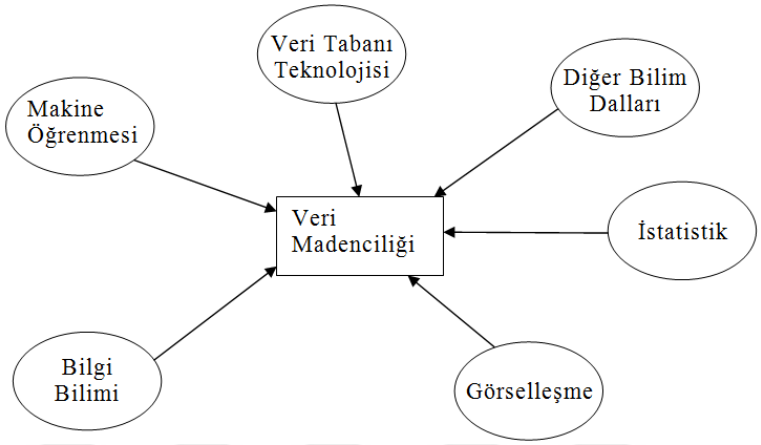
Şekil 2.12. Veri Ambarı Mimarisi

Günümüz kurum ve kuruluşlarında stratejik düzeydeki karar destek sistemlerinin yaratılması için gereksenen alt yapı, veri ambarı tarafından sağlanmaktadır. Dolayısıyla veri ambarı; kurum ve kuruluş içi ya da dışı, verilerin sorgu için hazır olmasını sağlamaktadır (Baykal, 2006).

2.3.2. Veri Madenciliği Süreci

Veri madenciliği; veri tabanı teknolojileri, istatistik, makine öğrenme, görselleşme, vb. alanları içermektedir (Şekil 2.13). Bu nedenle ilgili veriyi

elde etmek ve buna erişmek kolay olmayıp, bazı yöntemlerin uygulanması söz konusudur (Berry ve Linoff, 2000).

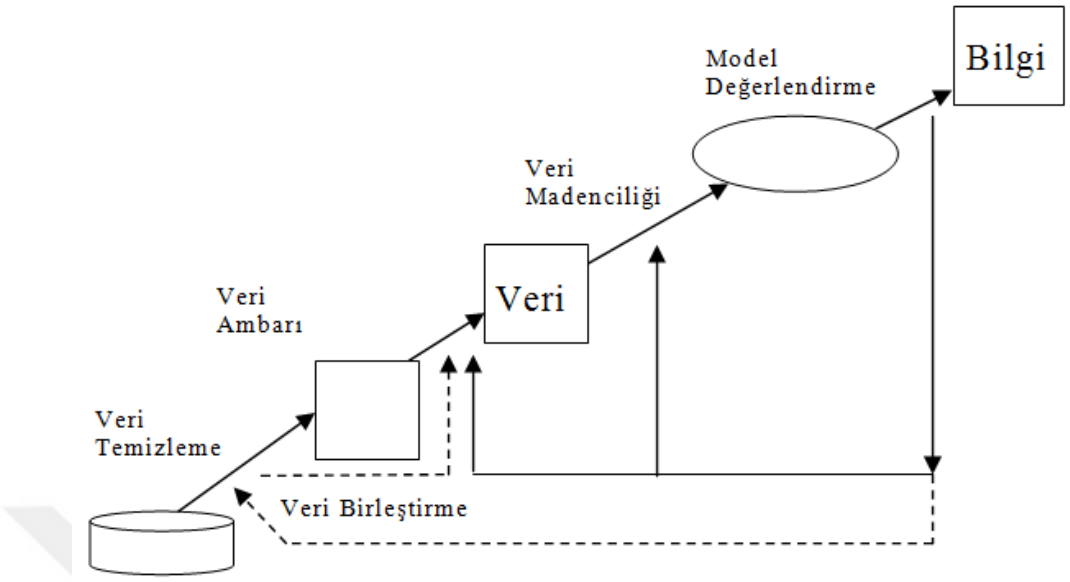


Şekil 2.13. Veri Madenciliği İlgi Alanları

(i). Bilgi Elde Etme Süreci

Veri madenciliği sistemi, belirli özellikleri taşıyan gerçek veriler üzerinde işlem yapmalıdır. Ancak ilginç ve değerli olan bilgiyi algılamak ve buna erişmek hiç de sanıldığı kadar kolay değildir.

Öncelikle konuya ilişkin amaçlar belirlenmelidir (Sever ve Ozel, 2006). Daha sonra analizi yapılacak verinin uygun veri tabanı belirlenerek, veri seçimi ya da elde edilecek alt veri örneklerinin oluşturulması sağlanmalı, gürültülü ve tutarsız veriler silinmelidir. Bu arada veri analizi ve veri dönüşümü yapılmalıdır. Yani analizde gerekli özelliklerin seçimi, özellikler arası ilişki belirleme, veri dönüşümü ve birleşimi yaparak boyut azaltılmaktadır. Bundan sonra veri madenciliği tekniği ve algoritması belirlenerek model değerlendirilmesi yapılarak bilgi sunumu ve yorumlaması gerçekleştirilmektedir (Şekil 2.14).



Şekil 2.14. Bilgi Elde Etme Süreci

(ii). Veri Madenciliği Modelleri

Veri madenciliğinde tahmin edici (predictive) ve tanımlayıcı (descriptive) olmak üzere iki temel yapı kullanılmaktadır. Tahmin edici modellerde, öncelikle sonuçları bilinen veriler kullanılarak bir model geliştirilmektedir. Daha sonra geliştirilen bu modelden yararlanılarak, sonuçları bilinmeyen veri kümelerine ilişkin sonuç tahmini amaçlanmaktadır. Tanımlayıcı modellerde ise, karar vermeye öncülük edecek verilerdeki yapıların tanımlanması sağlanmaktadır.

3. BÜYÜK VERİ (BIG DATA)

İçinde bulunduğumuz bilgi çağının gereği, veri/bilginin önemi ve yoğunluğu giderek artmıştır. Bilgi toplumunun unsuru olan akıllı telefon, bilgisayar ve bilgi teknolojileri yönetimi hayatımızın her alanına girmiş durumdadır. Dolayısıyla veri ve bilgi, yoğunluğu anlamlı ve nitelik kazanacak şekilde toplanmaya başlamıştır. Bunun sonucu veri ve bilgi artışına paralel olarak, bunlara erişim hızı da artmıştır. Veri/bilgideki niceliksel değişim, niteliksel değişimi de beraberinde getirmiştir. Veri/bilginin anlamlı bir bütün oluşturacak şekilde yoğun olarak toplanması, ilk olarak astronomi ve genetik bilim dallarında gerçekleştirilmiştir. Bu olgu, artık günümüzde hayatımızın her alanında kendini göstermeye başlamıştır.

Büyük veri, var olan bilgi sistemlerinin işleyemeyeceği kadar yoğun ve karmaşık veri kümeleri olarak tanımlanmaktadır. Başka bir deyişle, bilinen veri tabanı yönetim sistemleri ve yazılım unsurlarının, veri toplama, saklama ve çözümleme yeteneklerini aşan büyüklükteki veri kümelerine büyük veri denilmektedir. Günümüzde bu büyüklük terabyte'lardan, petabyte'lara (10^{15} byte) yükselmiştir.

Sosyal medya paylaşımları, ağ günlükleri, bloglar, fotoğraf, video, log dosyaları, vb. farklı kaynaklardan toplanan veriler geleneksel yapılarda saklanamaz boyuta ulaşmıştır. Söz konusu yoğun verinin de anlamlı ve işlenebilir hale dönüştürülmesi gerekmektedir. Dolayısıyla büyük veri; web sunucularının logları, internet istatistikleri, sosyal medya yayınları, bloglar, mikrobloglar, iklim algılayıcıları ve benzer sensörlerden gelen bilgiler ile GSM operatörlerinden elde edilen arama kayıtlarından oluşmaktadır (Demirtaş ve Argan, 2015).

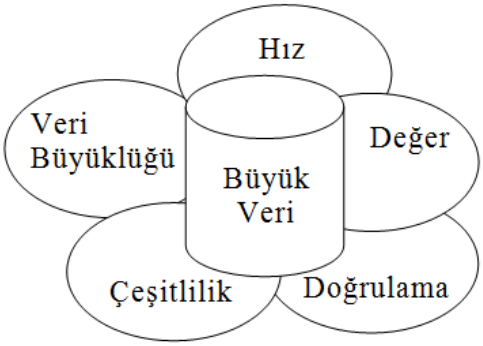
3.1. Büyük Veri Kavramı

1990'lı yılların ortalarından sonra gündeme giren büyük veri üzerinde yapılan çalışmalar göstermiştir ki, ortak bir tanımda birleşmek olası değildir (Hoy, 2014; Schönberger ve Cukier, 2014). Tanımlardan biri, büyük verinin büyüklüğüne ilişkindir. Örneğin terabit veya petabitin yüzlerce katı büyüklüğünde (Vinod, 2013). Verinin işlevselliği açısından yapılan tanımlama ise, değişik ortamlardan elde edilen verinin, veri madenciliği yöntemleriyle kullanılabilirliğini sağlamaktır (Rubinstein, 2013). Bir diğer tanım ise, verinin boyutu ve bunun bir süreç olduğu üzerinedir (Beyer ve Laney, 2012). Büyük veri tanımına ilişkin kavramsal bir yaklaşım ise, büyük verinin geleneksel yöntemler ile işlenmesi yerine daha değişik bir seçenek araştırılması gereğidir (Dumbill, 2013).

Nitelik ve nicelik açısından incelendiğinde, büyük veri hem olgu hem de süreçtir (Narayanan, 2014). Çünkü bir olgu olarak; büyük veri geleneksel veri saklama ve işleme yöntemleriyle işlenemeyen büyüklüklerden söz etmekte, süreç olarak da, büyük boyuttaki bu verinin analizi ve sonuç çıkarılmasını ifade etmektedir. Dolayısıyla büyük verinin, geleneksel yöntemlerle işlenemeyen veri değil de, elde edilen verinin yeni ve alternatif yöntemlerle işlenmesi süreci olarak ifade edilmesi daha uygundur. Büyük veri, oldukça büyük ve farklı veriyi kullanarak sonuç çıkarmayı amaçlayan yapıdadır. Büyük veriyi bilinen veri yapılarından ayıran özellikler; hızlı değişim, elde etme ve analizde tek model kullanmama, büyük hacim, hız ve çeşitlilik ile geleneksel saklama kapasitesini aşma olarak özetlenebilir. Bu nedenle büyük veri kullanımında, insan gücü ve teknolojik altyapı etkileşimi önemlidir.

3.2. Büyük Veri Bileşenleri

Büyük veri oluşumunda beş bileşen söz konusudur (Gobble, 2013). 5V olarak adlandırılan bu bileşenler, İngilizce karşılıklarından gelmektedir (Şekil 3.1).



Şekil 3.1. Büyük Veri Bileşenleri

(i). Çeşitlilik (Variety)

Elde edilen verinin büyük bir bölümü yapısal olmayıp, farklı kaynak ve teknolojileri içermektedir. Doğal olarak; telefon, tablet, mail sistemleri, fotoğraf, video ve bloglardan gelen verilerin çeşitliliği, yani homojen olmayan bu yapı yönetimi de zorlaştırmaktadır. Büyük veri sistemleri değişik kaynaklardan beslendiği için heterojen yapıya sahiptir.

(ii) Hız (Velocity)

Hızlı çoğalan veri, o veriye gereksinim duyan işlem sayısının ve çeşitliliğinin de aynı hızda artmasına neden olmaktadır. Dolayısıyla büyük veri, yüksek hızda bağlantı ve geniş bant büyüklüğü gerektirmektedir (Shaeffer ve Olson, 2014). Büyük verinin üretilme hızı yüksek olduğundan, doğal olarak işlev sayısı ve çeşitliliği de artmaktadır.

(iii). Veri Büyüklüğü (Volume)

Veri büyüklüğü ya da hacim, üretilen verinin büyüklüğünü ifade etmektedir (Hoy, 2014). Yapılan araştırmalar (IDC istatistikleri), 2020 yılında ulaşılacak veri miktarının 2009'un 44 katı olacağını belirtmektedir. Kurum ve kuruluşların bu kadar büyük veri ile nasıl bir çalışma yapılması gerektiğini kurgulamaları gerekmektedir (Göksu, 2014).

(iv). Doğrulama (Verification)

Bu kadar yoğun bir veri ortamında, veri akışının da güvenli olması söz konusudur. Dolayısıyla; veri akışı sırasında, doğru katmandan, olması gerektiği güvenlik düzeyinde izlenmesi, doğru kişiler tarafından görünebilir veya gizli kalması gerekmektedir (Göksu, 2014).

(v). Değer (Value)

Büyük veri olarak tanımlanan bu önemli olgunun, üretim ve işleme katmanlarından sonra kurum ya da kuruluş için artı değer yaratıyor olması gerekmektedir. Yani sistem, karar veriş süreçlerine etki ve doğru kararın verilmesi için hazır olmalıdır.

3.3. Büyük Veri Teknolojileri

Günümüz geleneksel yapıları bu verileri saklamak için yeterli değildir. İlişkisel veri tabanlarında veri bütünlüğü temel alındığı için büyük veri çözümlerine göre daha yavaştır. Ayrıca ilişkisel veri tabanlarında gigabyte düzeyinde işlem yapılırken, büyük veri için petabyte düzeyinden ve toplu işlem (batch processing) işlemlerinden söz edilmektedir. Büyük veri yaklaşımı, dağıtık dosya sistemine göre çalıştığı için veri bütünlüğünden söz edilemez. Yani ilişkisel veri tabanlarında olduğu gibi, şema ve ilişki

tablo yapısı yoktur. Çünkü büyük verinin tutarlılık (consistency), uygunluk (availability) ve parçalanma payı (partition tolerance) kurallarının hepsini sağlaması olası olmadığı için bazı verilerin doğru olmaması ya da kaybolması, veri büyüklüğü dikkate alınırsa çok da önemli değildir. Bu nedenle, büyük veriyi basit donanımların dağıtık dosya sistemleri ile birleşimi çözümler oluşturulmuştur (MapReduce, Hadoop, Storm, Hana ve NoSQL gibi). Bu veri çözümlerinden MapReduce, Google tarafından sorunları farklı birimlere bölerek hızlı işlemek için geliştirilmiştir. Günümüz sosyal ağlarından Facebook oldukça büyük Hadoop kümesine sahiptir. Bir diğer sosyal ağ Twitter, gerçek zamanlı veri işleme olanağı veren Storm'u geliştirmiştir. SAP tarafından geliştirilmiş olan Hana ise, verileri disk ortamında saklamak yerine ana bellekte daha hızlı işlemeye olanak sağlamaktadır. NoSQL (Not only SQL) ve Hadoop, günümüzde en yaygın olarak kullanılanlarıdır.

(i). NoSQL (Not Only SQL)

NoSQL, klasik ilişkisel veri tabanı yönetim sitemlerinden farklı olarak yatay ölçeklemeye göre veri saklama gerçekleştirmektedir. Bu nedenle klasik veri tabanlarını alt küme olarak görmektedir. 1998 yılında ortaya atılan bu kavram, ilişkisel modelden tamamen farklıdır. NoSQL, geleneksel yöntemlerin aksine yoğun oku/yaz işlemi gerçekleştirmektedir (örneğin sosyal ağlar).

NoSQL, mimarisi çoğu zaman tutarlı veya tek veri maddesiyle sınırlı işlemlerde zayıflık göstermesine karşın, yardımcı yazılım katmanları eklenerek sorun çözümlenmiştir. Örneğin Google'un Percolator ve Waterloo Üniversitesi'nde HBase için geliştirilmiş hareketsel sistemler gibi. Birbirinden bağımsız olarak geliştirilen bu sistemler, ara yazılım veya ara yazılım katmanından kaynaklanan ve bakıma gereksinim duymadan altındaki sütun gurubu için çok satırlı dağıtık işlemlerin yapılmasını sağlamaktadır.

Genel olarak NoSQL sistemi, verilerin farklı sunucularda yedeklemesini yapan dağıtık mimariyi kullanmaktadır. Bu işlem çoğunlukla dağıtık hash tablolarıyla yapılmaktadır. Dolayısıyla sistem kolayca yeni sunucular eklenerek büyütülebilmekte ve herhangi bir sunucunun arızalanması durumunda etkilenmemektedir.

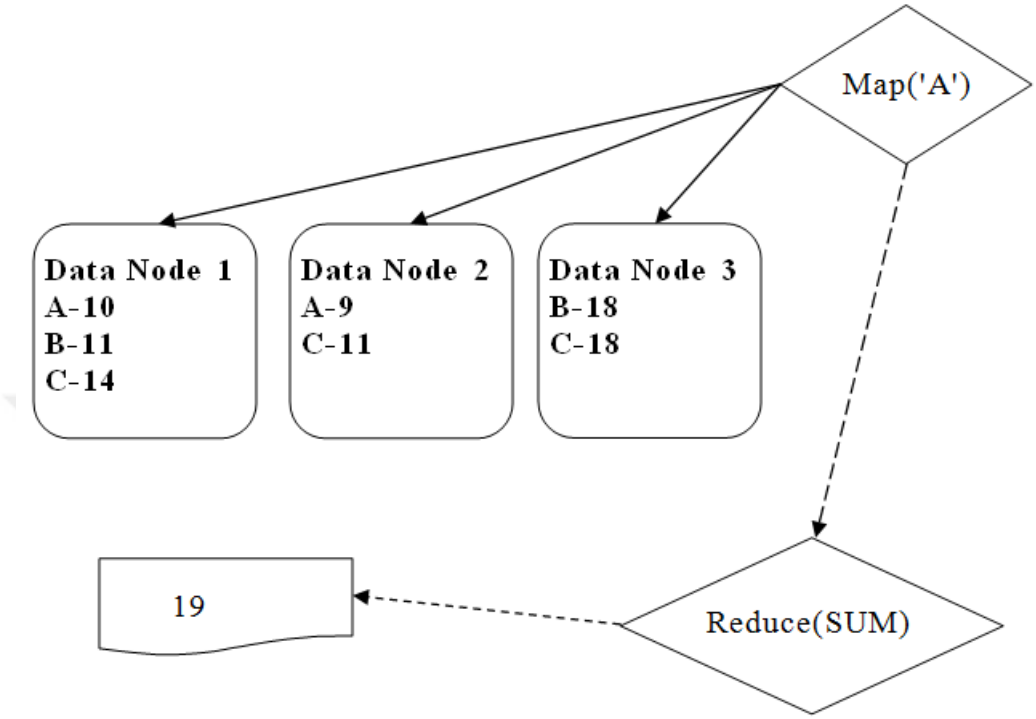
(ii). Hadoop

Apache Hadoop, büyük verinin farklı bilgisayarlarda eş zamanlı olarak analiz edilmesini sağlamaktadır. Analiz edilecek olan söz konusu veri, HDFS (Hadoop File System) üzerinde tutulmakta ve Hadoop diğer bilgisayarların oluşturduğu kümeler (cluster) üzerinde işlem yapmaktadır (Karaca, 2015). Bu yapı hem verinin hem de işlerin dağıtılmasını sağlamaktadır.

Apache Pig ve Apache Hive, SQL türü sorgular gerçekleştirilmesini ve bunun Hadoop işlerine dönüştürülmesini sağlamakta ve geliştirme aşamasını hızlandırmaktadır. Açık kaynak kodlu bir yazılım olan bu olgu, MapReduce algoritmasını soyutlayarak öğrenme eşiğini kolaylaştırmıştır. Apache Oozie ise, tanımlanan Hadoop işlerinin bir akış içinde belirlenen bir sırada ve aralıklarda çalıştırılmasını sağlamak için geliştirilmiştir.

MapReduce, dağıtık mimari üzerinde çok büyük verilerin kolay bir şekilde analiz edilebilmesini sağlayan bir sistemdir. Bu sistem, 1960'lı yıllarda geliştirilen fonksiyonel programlamadaki map ve reduce fonksiyonlarından esinlenilmiştir. Map aşamasında ana düğüm verileri alıp daha ufak parçalara ayırarak işçi düğümlere dağıtır. İşçi düğümler bu işlemleri tamamladıkça sonucunu ana düğüme geri gönderir. Reduce aşamasında ise tamamlanan işler işin mantığına göre birleştirilerek sonuç elde edilir. Başka bir deyişle Map aşamasında analiz edilen veri içerisinden almak istediğimiz veriler çekilir, Reduce aşamasında ise bu çektiğimiz veri

üzerinde istediğimiz analiz gerçekleşir. Açıklanan bu çalışma mantığı (Şekil 3.2)'de gösterilmiştir.



Şekil 3.2. MapReduce Çalışma Mantığı

3.4. Yararlar

Büyük veri olgusunun giderek daha da önem kazanacağı tahmin edilmektedir. Çünkü günümüz kurum ve kuruluşları, müşterilerinin günlük hatta anlık davranışlarını bilmek isteyeceklerdir. Daha önceleri bu mümkün olmayabilirdi. Ancak günümüzde sosyal ağların kullanımı arttıkça bu tür verilerin elde edilmesi de olanaklıdır. Doğal olarak bu verilerin çoğunluğu yapısal olmayan verilerdir. Dolayısıyla yapısal ve yapısal olmayan verinin saklanması, analizi ve veri madenciliği işlemlerine tabi tutulmasının giderek önem kazanacağı yadsınamaz bir gerçektir.

Günümüzde veri, tüm endüstrinin ve üretimin vazgeçilmez bir parçası haline gelmiştir. Dolayısıyla veri ve veri bilimi yeni istihdam alanları

yaratmaktadır. Büyük veriden değer yaratmak için; veriyi sıklıkla, daha şeffaf ve kullanışlı hale getirerek kuruluş için önemli olan değerler ortaya çıkarılmalıdır. Kuruluşlar bu şekilde daha işlevsel veriler yarattıkça ve bunları etkin şekilde sakladıkça performans artışı sağlanacaktır. Büyük veri çok daha özelleştirilmiş pazar bölümlemesi sağlayacağından ileriye dönük tahmin, yeni ürün ve hizmetlerin geliştirilmesine katkı sağlanmış olacaktır. Küçük ve orta boy kuruluşlar, içinde bulundukları rekabetçi ortamda daha güçlü olabileceklerdir. Büyük veri sayesinde tüketiciler kendilerine özgü mal ve hizmetlere kolay ve hızlı olarak ulaşabileceklerdir. Karmaşık analizler ve karar verme etkin ve bir o kadar da kolaylaşacaktır.

4. UYGULAMA

4.1. Bulut Programlama (Cloud Computing)

Bulut Programlama (Cloud Computing) veya işlevsel anlamıyla çevrim içi bilgi dağıtımı; bilişim aygıtları arasında ortak bilgi paylaşımını sağlayan hizmetlere verilen genel addır. Bulut bilişim bu yönüyle bir ürün değil, hizmettir. Temel kaynaktaki yazılım ve bilgilerin paylaşımı sağlanarak, mevcut bilişim hizmetinin bilgisayarlar ve diğer aygıtlardan elektrik dağıtıcılarına benzer bir biçimde bilişim ağı (genellikle internet) üzerinden kullanılmasıdır.

4.2. Amazon Web Services (AWS)

Amazon.com, dünyanın bir numaralı e-ticaret mağazası olarak bilinmektedir. Kabul edilmelidir ki, bu kadar büyük bir platformu yürütmek ve işletmek önemli bir altyapı yatırımı gerektirmektedir. Amazon Web Services (AWS), dünyanın değişik bölgelerinde veri merkezleri bulunan bulut bilişim ve depolama hizmeti sunan Microsoft ve Google gibi büyük bir servis sağlayıcıdır (Wikipedia Amazon WebSevices, 2014).

(i) Amazon Web Services (AWS) Kavramı

AWS çevrim içi hizmetler sunmaya 2006 yılında başlamış olup, dünyanın değişik bölgelerinde veri merkezlerine sahiptir (Şekil 4.1). Her bölge, kullanılabilirlik bölgeleri olarak adlandırılan çok sayıda daha küçük coğrafi bölgeleri kapsamaktadır. Coğrafi olarak dağıtılmış AWS; kesintileri önleme, hizmetleri yedekleme ve onları yüksek oranda kullanabilir hale getirmeyi amaçlamaktadır.

Şekil 4.1. AWS Veri Merkezi Bölgeleri Ekran Görüntüsü

Amazon web services (AWS), erişimi birkaç dakika içerisinde çok kolay olan bilgisayar kaynakları hizmeti sunmaktadır. Bunun yanında 'kullandığın kadar öde' sistemi ile fiyatlandırma yapmaktadır. Bu da müşterilere, kiraladıkları AWS hizmetini kullanmadıkları zaman sürecinde ödeme yapmama fırsatı vermektedir. Dolayısıyla bu durum maliyetleri daha hesaplı hale getirmektedir. Örneğin, bu tez çalışması kapsamında Linux makine düğümleri çalıştırıldıkları saat başına 0.06$ olarak fiyatlandırılmıştır.

(ii). Amazon Web Services (AWS) Yararları

Amazon'un belirttiği AWS kullanımının yararları;

- Düşük maliyet
- Satın alma maliyetinin olmaması
- Esneklik
- Ölçeklenebilirlik
- Yüksek kullanılabilirlik
- Güvenlik

olarak özetlenebilir.

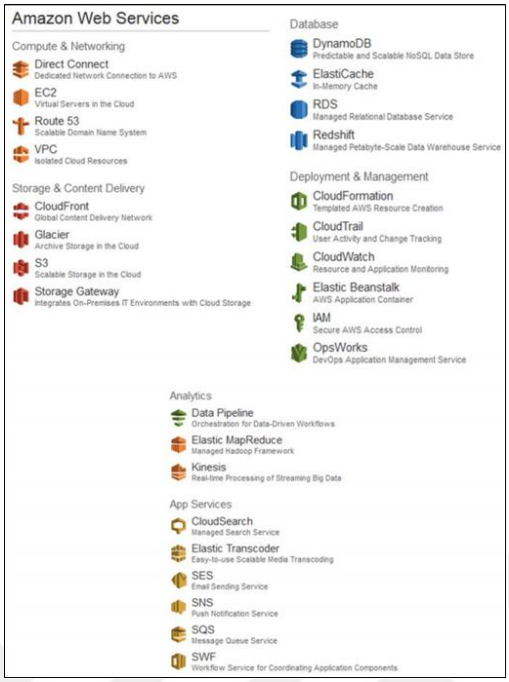
4.3. Elastic MapReduce (EMR)

Amazon Elastic MapReduce (Amazon EMR); büyük miktardaki veriyi kolay, hızlı ve düşük maliyetle işlemek amacıyla Amazon müşterilerine sunulmuş bir web servisidir. Amazon EMR, verileri dağıtmak için Hadoop kullanmakta ve boyutlandırılabilir Amazon EC2 örnekleri kümesinde işlem yapmaktadır.

4.3.1. EMR Kavramı

Amazon EMR kümesini oluşturmak oldukça kolaydır. Bu küme oluşturulduğunda, diğer Amazon web hizmetleri ile çalışabilir halde düzenlenmiş, Hadoop API düğümü ve Hadoop için tablo sorgulamada kullanılan bir yapıda kullanıma sunulmaktadır. Bu yapı ve hazır kurulumlar, diğer kurulum gereksinimleri hakkında kullanıcıyı endişelendirmeden veri analizi üzerinde çalışmayı kolaylaştırmaktadır.

Amazon EMR, birçok küme kurmak veya düğüm oluşturma olanağı sağlamaktadır. Söz konusu bu elemanları oluşturma mönüsü (Şekil 4.2)'de gösterilmiştir (Amazon Elastic MapReduce, 2009 - 2014). AWS, aynı zamanda başarı ile tamamlanamamış görevlerin de görüntülenmesini desteklemektedir.



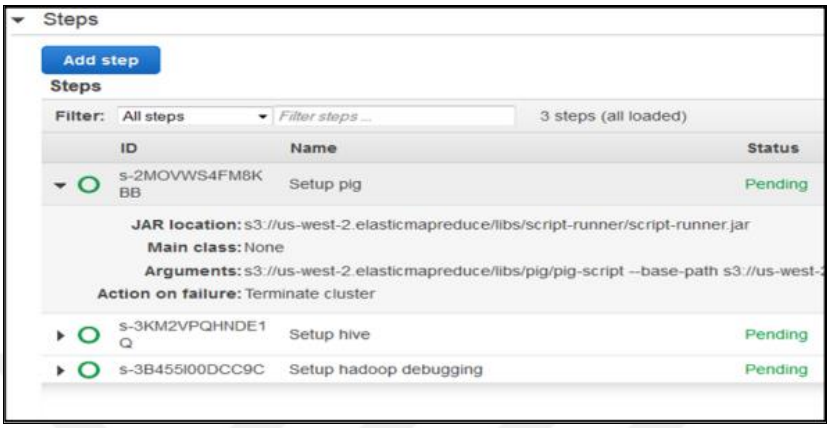
Şekil 4.2. Amazon EMR Elemanları Oluşturma Menüsü Ekranı

4.3.2. EMR Terminolojisi

Kavramsal olarak açıklanan EMR terminolojisi aşağıda belirtilmiştir.

- Küme
- Düğümler
    - Ana düğüm: Ana düğüm kümeyi yönetmektedir. Yani ham veri dağılımının haritasını tutar ve çalıştırılan her görevin ve örneklerinin durumunu izler. Bir kümede sadece tek bir ana düğüm bulunmaktadır.
    - Çekirdek düğümler: Dağıtık dosya yönetim sisteminde (HDFS) bulunan dağıtık veriyi tutan işçi düğümlerdir.
    - İşçi düğümleri: İşleri dağıtma yeteneğini sağlamak için AWS üzerinde etkisi olan ek işçi düğümlerdir.
- Adımlar, yapılan birim işler olarak tanımlanmaktadır. AWS'nin işleyebileceği en fazla adım sayısı 256'dır. Buna ek olarak (Şekil 4.3)'de görüldüğü gibi AWS, kullanıcıya adımların durumunu

görüntüleme olanağı da sunmaktadır (Amazone Elastic MapReduce, 2009 - 2014).



Şekil 4.3. AWS Hadoop Kümesi Adım İzleme Sayfası Ekran Görüntüsü

4.4. Elastic Computing Cloud (EC2)

Çeşitli işletim sistemlerine, uygulama olanağı veren programlama ortamıdır. Oluşturulmuş örneklerden seçmek ve çalıştırmak için kullanıcı ara yüzü ile birlikte sunulmaktadır. Örneğin, var olan erişim yönetim ara yüzü yardımı ile farklı yazılımlar veya uygulamalar ile özelleştirilebilmektedir.

EC2 özellikleri;

- AMI adı verilen önceden konfigüre edilmiş şablon desteği,
- Herhangi bir sayıda örnek AMI tarafından çalıştırılabilme,
- Örnekler için işlemci, bellek ve depolama çeşitliliği,
- Genel veya özel anahtar ayrımı ile sisteme güvenli giriş özelliği,
- EC2 üzerinde kalıcı ve geçici depolama birimlerinin olabilirliği,
- Dinamik bulut programlama için esnek IP adresi,

şeklinde özetlenebilir (Amazon Elastic Compute Cloud, 2014).

4.5. Veri Depolama Servisi (S3)

Amazon, bulutta veri depolamak için bir depolama servisi sunmaktadır. Veriler indirilebilmekte veya farklı AWS servisleri ile kullanılabilmektedir. Amazon, S3 kovaları adı verilen yapı ile verileri saklamaktadır. Kovalar, alan adlarına benzemektedir. (s3.amazonaws.com)'un S3 kovaları ve alt alan adları doğrudan eşleşmektedir.

4.6. AWS'ye Hadoop Kurulumu

Bu çalışma kapsamında, Hadoop kümesi üzerinde performans analizi yapılmıştır. Bu çalışma öncesinde, sistemin söz konusu çalışma için hazır hale getirilmesinde aşağıdaki adımlar izlenmiştir.

1. http://aws.amazon.com/ adresinden AWS için kayıt olunmuş ya da daha önce Amazon hesabı bulunuyorsa onun yardımı ile sisteme giriş yapılmıştır.
2. Log dosyaları ve data için AWS S3 üzerinden kova oluşturulmuştur.
3. Daha sonra da erişim anahtarı oluşturulmuştur.
    a. EMR kümesi oluşturulurken EC2 düğümleri çalıştırılır ve bu düğümlere ulaşmak için bir erişim anahtarına ihtiyaç duyulur. Bu anahtar; https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#KeyPairs: adresinden, pem formatında indirilmektedir.
    b. .pem uzantılı anahtarı genel ve özel anahtar şeklinde ayırmak için PUTTYGEN uygulamasına gereksinim duyulmaktadır. PUTTYGEN uygulaması çalıştırıldıktan sonra, EC2 ile oluşturulan .pem uzantılı anahtar .ppk uzantılı hale dönüştürülür ve diskte saklanır.

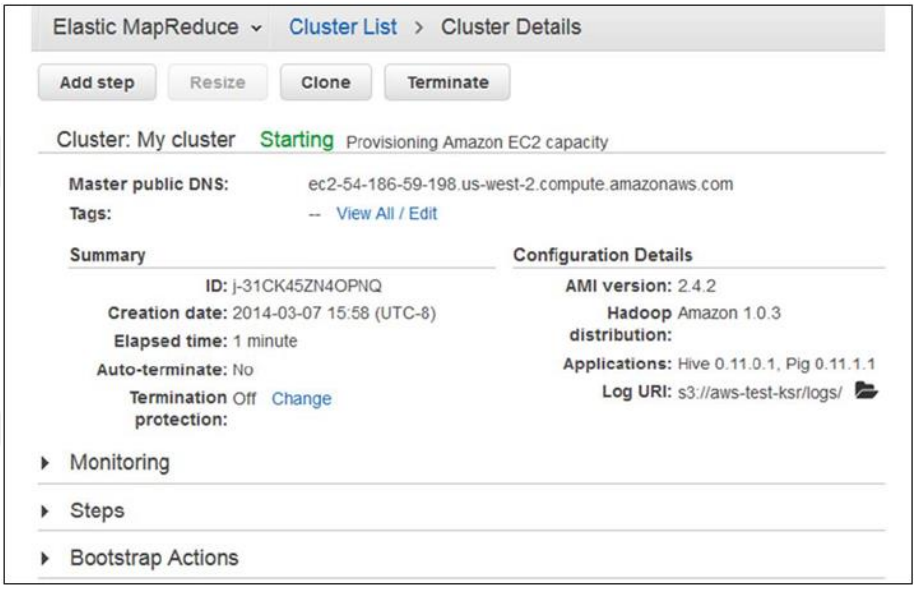
Küme oluşturma işlemi (Şekil 4.4)'de gösterilmiştir.

Şekil 4.4. Küme Düzenleme Sayfası Ekran Görüntüsü

c. AWS yönetim konsol menüsünden AWS EMR seçilir.
d. Küme oluşturma butonuna tıklanır.
e. Küme adı girilir.
f. Loglama için klasör seçilir.
g. Yapılan çalışma için seçili olarak gelen son sürüm Hadoop seçeneği ile devam edilir.
h. Donanım düzenlemesi seçeneklerinde ana düğüm bir, çekirdek düğüm iki ve işlemci düğüm sıfır olarak seçilir.
i. Güvenilir ve erişim bölümünde daha önce oluşturulan AWS anahtarı seçilir ve "Küme Oluştur" butonuna basılarak istenilen küme oluşturulmuş olur.
j. Böylece EC2 üzerinde bir tane ana, iki tane de işçi düğüm çalışır hale getirilmiş olur.
k. Kümedeki bütün düğümlere (Şekil 4.5)'de belirtilen şekilde genel DNS adı atanır.

4. Ana kümeye erişim için öncelikle ana düğümün dosya sistemine veriler yüklenir ve hive veri tablosuna kopyalanır. Bu işlem için;
   a. Windows işletim sistemi için WinScp uygulaması yüklenir.

b. WinScp ile ana düğüme erişim, (Şekil 4.6)'da belirtilen şekilde ana düğümün DNS adı kullanılarak gerçekleştirilmektedir; kullanıcı adı 'hadoop'.
c. WinScp uygulamasından 'Gelişmiş' butonuna tıklanır ve SSH->Kimlik Denetleme seçeneğindeki, daha önce kaydedilmiş olan anahtar seçilir.
d. Ok butonuna tıklanır ve erişim sağlanmış olur.

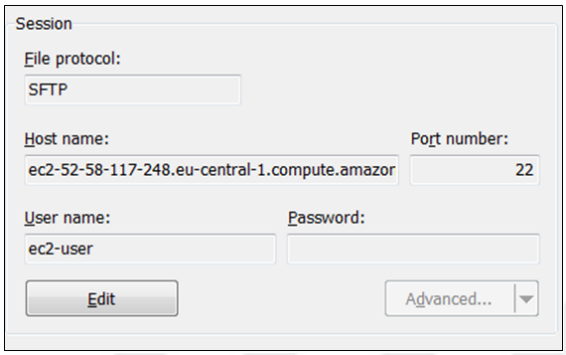


Şekil 4.5. Küme Özellikleri Görünüm Sayfası Görüntüsü

e. Bu şekilde ana düğümün dosya sistemine erişilmiş olur.

5. Hadoop/Hive komutlarının çalıştırılabilmesi için ana düğümün terminaline SSH ile bağlanılması gerekmektedir. Bunun için;
   a. Putty uygulaması çalıştırılır,
   b. Bağlanılacak bilgisayar adı girilir (örneğin hadoop@masternodeDNS),
   c. Category bölümünden Connection->SSH->Auth altına önceden kaydedilmiş anahtar seçilir,
   d. 'Open' butonuna tıklanarak bağlantı sağlanmış olur.

6. Belirtilen bu işlem adımları sonucunda; AWS üzerine Hadoop kümesi kurulmuş, dosya sistemine ulaşılmış ve ana düğüm terminalinde komut çalıştırmak için bağlantı sağlanmış olmaktadır.



Şekil 4.6. WinScp Ana Düğüm DNS Adlı Ekran Görüntüsü

5. PERFORMANS ANALİZİ

Çalışmanın bu aşamasında, Hadoop dağıtılmış paralel programlama ile geleneksel ilişkisel veri tabanı yönetim sistemleri arasındaki sorgu çalıştırma zamanları karşılaştırılmıştır. Yapılan çalışmada performans, artan veri seti büyüklüğüne göre değerlendirilmiştir.

5.1. Örnek Veri Seti

Çalışmada kullanılan veri setinin özellikleri aşağıda belirtilmiştir.

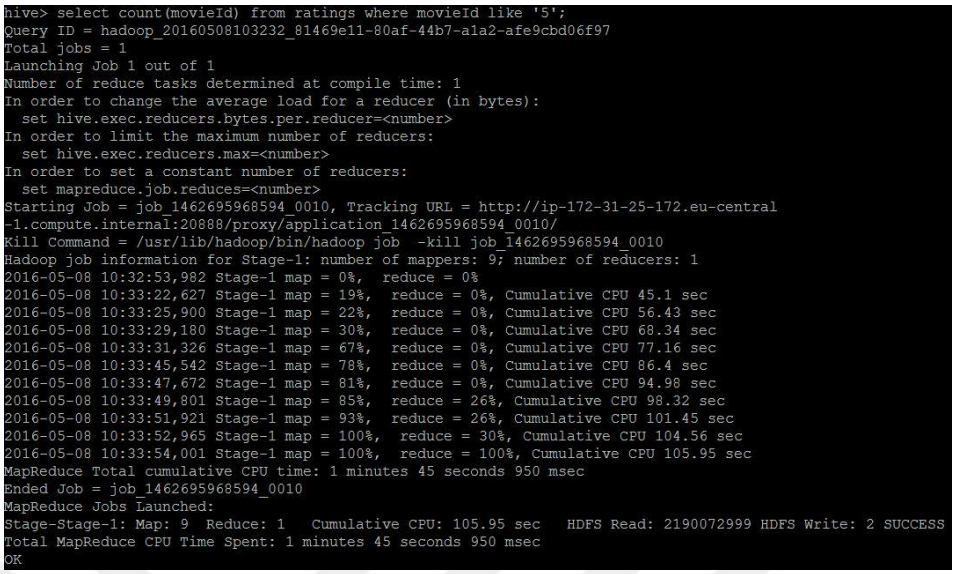
- Minesota Üniversitesi tarafından toplanan veri seti kullanılmıştır.
- Bu çalışmada MovieLens 10M veri seti kullanılmıştır.
- Veri CSV formatında, 72.000 kullanıcı tarafından 10.000 film hakkında 10 milyon değerlendirme içermektedir.
- (Tablo 1)'de veri şeması gösterilmiştir.

Kolon Adı	Veri Tipi
userId	metin
col1	metin
movieId	metin
rating	metin

Tablo 1. WinScp Ana Düğüm DNS Adlı Veri Şeması Düzeni

5.2. Hive

Hive, SQL formatında büyük dağıtık veri setlerini sorgulamak amaçlı Apache tarafından sağlanan açık kaynak kodlu bir veri ambarı yazılımıdır. Hive, veriyi yapılandırılmış şekilde canlandırmaya yaramakta ve HiveQL adı verilen SQL gibi bir sorgu dilini desteklemektedir. (Şekil 5.1)'de Hive ile çalışan sorgunun sonucu gösterilmiştir.



Şekil 5.1. Hive İle Çalıştırılan Örnek Sorgu Ekran Görüntüsü

5.3. Deneyler

Gerçekleştirilen deneyler;

- Hadoop kümesi veri yükleme performansı,
- Hadoop kümesinde sorgu çalıştırma performansı,
- Geleneksel veri tabanı yönetim sistemi ve Hadoop veri yükleme performansı analizi,
- Geleneksel veri tabanı yönetim sistemi ve Hadoop sorgu çalıştırma performansı analizini,

içermektedir.

5.4. Geleneksel Veri Tabanı Yönetim Sistemi ve Hadoop Arasında Sorgu Çalıştırma

Bu çalışmada, geleneksel veri tabanı sistemleri ile hadoop arasındaki sorgu çalıştırma zamanları karşılaştırılmıştır. Çalışma, 4 GB ve 6 GB büyüklüklerindeki veri setleri üzerinde gerçekleştirilmiştir. Çalışmanın sonuçlarının karşılaştırılması için Minesota Üniversitesi tarafından ücretsiz olarak kullanıma sunulan bir veritabanından yararlanılmıştır. Veritabanı 4 GB ve 6 GB boyutlarında olmak üzere iki veri ayrı seti olarak

hazırlanmıştır. Öncelikle bu veri setleri geleneksel bir veri tabanı yönetim sistemi üzerine aktarılmıştır. Dolayısıyla test ortamı olarak aynı özelliklere sahip bilgisayarlarda, (Tablo 2)'de gösterilen sorgular ayrı ayrı çalıştırılarak süre ölçümleri yapılmıştır.

Sorgu No	Sorgu
S1	Select * from ratings
S2	Select movieId from ratings
S3	Select * from ratings order by userId
S4	Select count(*) from ratings
S5	Select count(*) from ratings group by userId
S6	Select * from ratings where ratings like ‘5’
S7	Select movieId from ratings where ratings like ‘5’
S8	Select count(*) from ratings where ratings like ‘5’
S9	Select max(ratings) from ratings
S10	Select min(ratings) from ratings
S11	Select average(rating) from ratings
S12	Select average(rating) from ratings group by movieId
S13	Select * from ratings order by userId asc
S14	Select * from ratings order by userId desc
S15	Select * from ratings order by rating
S16	Select count(*) as nbratings, rating from ratings group by rating order by rating desc
S17	Select count(userid) as nbusers, avgrating from (select round(avg(rating),1) as avgrating, userId from ratings group by userId) as avgratingbyusers group by avgrating order by avgrating desc
S18	Select useId, col1, movieId, count(*) as sumRows From (SelectuseId, col1, movieId, row_number() over (Partition ByuseId) as row from ratings) rs Where row <= 10000 Group ByuserId, col1, movieId
S19	Select * From(Select userId from ratings Union All Select movieId from ratings) rn
S20	Select * From (Select userId from ratings Union All Select movieId from ratings) rn Order by userId

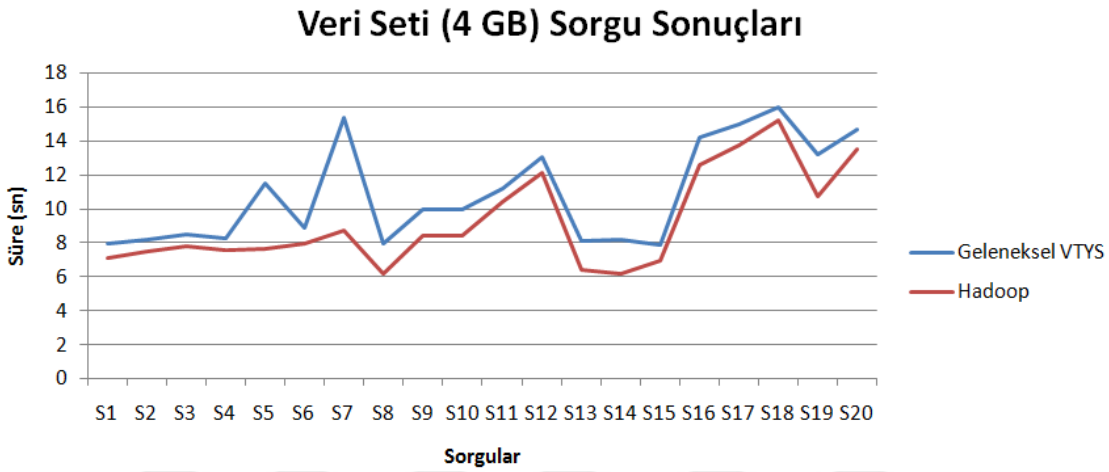
Tablo 2. Hadoop ve Geleneksel Veri Tabanı İçin Çalıştırılan Sorgular

Oluşturulan sorguların aynı özeliklere sahip bilgisayarlarda çalıştırılması ile elde edilen sonuçlar (Tablo 3)'de gösterilmiştir. Söz konusu sorgular farklı boyutlardaki veri setleri üzerinde çalıştırılmıştır.

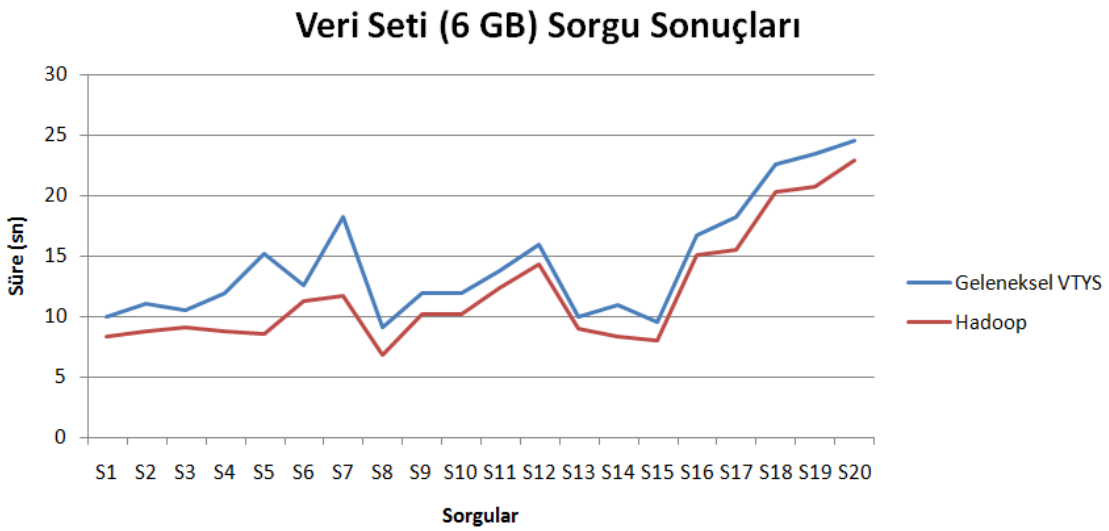
Veri Seti Büyüklüğü	Sorgu No	Geleneksel VTYS (sn)	Hadoop (sn)
4 GB	S1	8	7.1
	S2	8.2	7.5
	S3	8.5	7.8
	S4	8.3	7.6
	S5	11.54	7.65
	S6	8.9	8
	S7	15.4	8.7
	S8	8	6.2
	S9	10	8.4
	S10	10	8.4
	S11	11.2	10.4
	S12	13.1	12.1
	S13	8.1	6.4
	S14	8.2	6.2
	S15	7.9	7
	S16	14.2	12.6
	S17	15	13.7
	S18	16	15.2
	S19	13.2	10.7
	S20	14.7	13.5
6 GB	S1	10	8.4
	S2	11.1	8.8
	S3	10.5	9.2
	S4	12	8.8
	S5	15.2	8.6
	S6	12.6	11.3
	S7	18.2	11.7
	S8	9.14	6.88
	S9	12	10.2
	S10	12	10.2
	S11	13.8	12.4
	S12	16	14.3
	S13	10	9
	S14	11	8.4
	S15	9.6	8.1
	S16	16.7	15.1
	S17	18.2	15.5
	S18	22.5	20.3
	S19	23.4	20.7
	S20	24.5	22.9

Tablo 3. Sorguların Çalıştırılması İle Elde Edilen Sonuçlar

Veri seti büyüklüğü 4 GB olan veriler üzerinde çalıştırılan sorgulardan elde edilen sonuç grafiği (Şekil 5.2)'de ve veri seti büyüklüğü 6 GB olan veriler üzerinde çalıştırılan sorgulardan elde edilen sonuç grafiği ise (Şekil 5.3)'de gösterilmiştir.



Şekil 5.2. 4 GB Veri Seti Sorgu Sonuçları



Şekil 5.3. 6 GB Veri Seti Sorgu Sonuçları

Sorgu karmaşıklığı ve kullanılan veri setinin büyüklüğü, sorgu sürelerini etkileyen bir faktördür. Elde edilen sonuç grafikleri göz önünde bulundurulduğunda, sorgu uzunluğu kısa olan sorgularda her iki veri setinde de süreler oldukça kısadır. Buna karşılık sorgu uzunluğunun daha büyük olduğu sorgularda, sorgu sonuçlanma sürelerinin daha uzun süre aldığı gözlemlenmiştir. Ayrıca geleneksel veritabanı yönetim sistemlerinde, aynı sorguların sonuçlanma sürelerinin nispeten daha fazla zaman aldığı görülmüştür. Hadoop gibi dağıtık veri kümeleri üzerinde yapılan sorgularda veri seti büyüklüğü ve sorgu karmaşıklığı ne olursa olsun, sorguların geleneksel veri tabanı yönetim sistemlerine göre daha kısa sürede sonuçlandığı tespit edilmiştir. Bu durum büyük veriler üzerinde sorgu, analiz vb. işlemlerin daha kolay gerçekleştirilmesine olanak sağlamaktadır.

6. SONUÇ

Büyük veri, kuruluşların günlük aktiviteleri arasında önemli bir yere sahip olmaya başlamıştır. Dolayısıyla büyük veri teknolojisinin kısa zaman içinde, bütün işletmelerin kullanacakları yeni nesil bir teknoloji olacağına kesin gözüyle bakılmaktadır.

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

Hadoop, kolayca ölçeklenebilir bir ortamda büyük veri analizi açısından oldukça kabul gören ve kullanılan açık kaynak kodlu bir yazılımdır. Ayrıca Hadoop; güvenilir, düşük maliyetli, dağıtılmış paralel programlamayı destekleyen, yapılandırılmamış veriyi saklama ve analiz edebilme özelliklerine sahiptir. Bunun sonucu olarak da sektörün öncü kuruluşları olan Google, Yahoo ve Facebook tarafından tercih edilmektedir.

Hadoop'un önceki sürümlerinde gerçek zamanlı veri analizi bileşeni yoktu, ancak yakın geçmişte gerçek zamanlı büyük veri analitiği için Apache Spark'ı tanıtmıştır. Spark'ın, esnek halde dağıtılmış veriye dayandığı ve sonuçlarını saniyeden daha kısa bir sürede verdiği belirtilmektedir.

Gelecekte, gerçek zamanlı büyük veri analitiği motoru geliştirmenin yararlı ve ilginç olacağı kaçılmazdır.

KAYNAKLAR

1. Referans Kaynaklar

Amazon Elastic Compute Cloud. Instances and AMIs, 2014. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html, accessed April 2014

Amazon Elastic MapReduce. What is Amazon EMR?, 2009. http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-isemr.html, accessed April 2014.

Andrews, M., (1987), Computer Organization, Computer Science Press, USA.

Bartee, T., (1985), Digital Computer Fundamentals, McGraw-Hill Book Co., New York, USA.

Baykal, A., (2006), Veri Madenciliği Uygulama Alanları, D.Ü.Ziya Gökalp Eğitim Fakültesi Dergisi, 7: 95-107.

Berry, M.J. ve Linoff, G.S., (2000), Mastering Data Mining, John Wiley &Sons Inc., New York.

Beyer, M.A. ve Laney, D., (2012), The importance of “big data” : A defination, Gartner Report, https://www.gartner.com/doc/2057415/importance-big-data-definition.

Çölkesen, R., (2004), Veri Yapıları ve Algoritmalar, Papatya Yayıncılık, İstanbul, Türkiye.

Davis, G., (1981), Computer Data Processing, McGraw-Hill Book Co., Tokyo, Japan.

Demirtaş, B. ve Argan, M., (2015), Büyük Veri ve Pazarlamadaki Dönüşüm : Kuramsal Bir Yaklaşım, Pazarlama ve Pazarlama Araştırmaları Dergisi, 15 (1-21).

Dumbill, E., (2013), Making Sense of Big Data, Big Data : 1 – 2.

Gobble, M.A.M., (2013), Big Data : The Next Big Think in Innovation, Research Technology Management, January-February : 64 – 66.

Gonnet, G.H., (1984), Handbook of Algorithms and Data Structure, Addision-Wesley Pub.

Göksu, C., (2014), Datawarehouse Türkiye, http://datawarehouse.gen.tr/big-data-nedir-geleneksel-veri-yonetimine-etkisi-ne-olur/

Hoare, C.A.R., (1975), Recursive Data Structures, International Journal of Computer and Information Sciences 4, 105-32.

Horowitz, E. ve Sahni, S., (1982), Fundamentals of Data Structures, Computer Science Press Inc., Rocville, Maryland, USA.

Horowitz, E. ve Sahni, S., (1976), Fundamentals of Computer Algorithms, Pitman Publishing Ltd., London, England.

Hoy, B., (2014), Big Data : An Introduction for Librarians, Medical Reference Servicer Quarterly, 33(3) : 320 – 326.

Karaca, İ., (2015), Büyük Veri Analizlerinin Kurumsal Faaliyetlerde Kullanım Alanları, Ankara Üniversitesi DTCF (Tez).

Kaşlı, A., (2003), Algoritma Tasarımı ve Programlama, Ege Üniversitesi Basımevi, Bornova, İzmir.

Kaya, H. ve Köymen, K., (2008), Veri Madenciliği Kavramı ve Uygulama Alanları, Doğu Anadolu Bölgesi Araştırmaları, 159-164.

Keskinel, F., (1985), Veri Tabanı Kavramı, Enka Yayınları, İstanbul, Türkiye.

Kurnaz, S., (2004), Veri Yapıları ve Algoritma Temelleri, Papatya Yayıncılık, İstanbul, Türkiye.

Martin, J., (1977), Computer Data-Base Organization, Prentice-Hall, Inc., Englewoods Cliffs, New Jersey, USA.

Narayanan, V., (2014), Using Big Data Analytics to Manage Data Deluge and Underlock Real Time Business Insights, Journal of Equipment Lease Financing, 32(2) : 1-7.

Özkan, Y., (2006), Veri Ambarı, Türkiye Bilişim Ansiklopedisi, 879-883.

Özkan, Y., (2009), Veri Tabanı Sistemleri, Alfa Basım Yayım Dağıtım Ltd. Şti.

Sever, H. ve Ozel, B.O., (2006), Veri Madenciliği, Türkiye Bilişim Ansiklopedisi, 883-887.

Schaeffer, D.M. ve Olson, P.C., (2014), Big Data Options for Small and Medium Enterprises, Review of Bsiness Information Systems, 18(1) : 41 – 46.

Schönberger, V.M. ve Cukier, K., (2014), Big Data – A Revolution That Will Transform How We Live, Work and Think, John Muray Publishers, London.

Rubinstein, I.S., (2013), Big Data : The end of privacy or a new beginning? , International Data Privacy, 3(2) : 74 – 86.

Tremblay, J. ve Sorenson, P., (1984), An Introduction to Data Structures with Applications, McGraw-Hill Book Co., New York, USA.

Vinod, B., (2013), Leveraging Big Data for Competitive Advantage in Travel, Journal of Revenue and Pricing Management, 12(1) : 96 -100.

Wikipedia, Amazon Web Services, 2014.
en.wikipedia.org/wiki/Amazon_Web_Services, accessed April 2014.

Yarımağan, Ü., (2000), Veri Tabanı Sistemleri, Akademi-Bilişim Vakfı.

2. Diğer Kaynaklar

Baase, S. ve Gelder, A.V., (2000), Computer Algorithms Introduction to Desing and Analysis, Addison Wesley Longman.

Bradley, J., (1981), File and Data Base Techniques, Tolt, Rinehart and Winston.

Date, C.J., (1990), An Introduction to Database System, Addison-Wesley Publishing Company Inc., USA.

Elmasri, R. ve Navathe, S.B., (2007), Fundamentals of Database Systems, Addison-Wesley.

Ergen, M.Ö., (1990), Veri Seti Düzenleme, Ege Üniversitesi Basımevi, İzmir, Türkiye.

Hanson, O., (1982), Design of Computer Files, Pitman Publishing Ltd., London, England.

https://tr.wikipedia.org/w/index.php?title=Büyük_veri&oldid=16498357

Humphries, M., Hawkins, M.W. ve Dy, M.C., (1998), Datawarehousing, Architecture and Implementation, Prentice Hall.

Imhoff, C., Galemmo, N. ve Geiger, J.G., (2003), Mastering Data Warehouse Design-Relational and Dimensional Techniques, Wiley.

Inmon, W.H., (2002), Buijding the Data Warehouse, Wiley.

Kalıpsız, O., (2001), Bilgisayar Veri Tabanı Sistemleri, Der Yayınları, İstanbul, Türkiye.

Kimball, R., (2002), The Data Warehouse Toolkit, Wiley.

Kruse, R.L., Leung, B.P. ve Tondo, C.L., (1991), Data Structures and Program Design in C, Prentice-Hall.

Lewis, T.G. ve Smith, M.Z., (1976), Applying Data Structures, Houghton Mifflin Co., Boston, USA.

Mader, C. ve Hagin, R., (1974), Information System : Technology, Economics, Applications, Science Research Associates, Inc., Chicago, USA.

Mattison, R., (1996), Data Warehousing: Strategies, Technologies and Techniques, McGraw-Hill.

McFadden, F.R. ve Hoffer, J.A., (2005), Modern Database Management, Prentice-Hall.

Sağıroğlu, S. ve Sinanç, D., (2013), Big Data: A Review, 2013 International Conference on Collaboration Technologies and Systems (CTS), 42-47.

Sahni, S., (1997), Data Structure: Algorithms and Applications in C++, McGraw Hill.

Setty, K. ve Bakhshi, R., (2013), What Is Big Data and What Does It Have to Do With IT Audit?, ISACA Journal, 3:23-25.

Shaffer, C.A., (1997), A Practical Introduction to Data Structure and Algorithm Analysis, Prentice-Hall, USA.

Standish, T.A., (1998), Data Structures in Java, Addison-Wesley, Reading, Massachusetts, USA.

Ponniah, P., (2002), Data Warehousing Fundamentals: A Comprehensive Guide fot IT Professionals, Wiley Interscience.

Powell, G., (2006), Beginning Database Design, Wiley Pub.

Tenenbaum, A. Ve Augensteins, M.J., (1981), Data Structures Using Pascal, Prentice-Hall.

Tenenbaum, A.M., Langsam, Y. ve Augenstein, M.J., (1990), Data Structure Using C, Prentice-Hall.

Wiederhold, G., (1988), File Organization fot Database Design, McGraw-Hill Book Co., Singapore.

Zorlu, E., Big Data, https://emrezorlu.com/2014/07/12/big-data/, erişim tarihi: Nisan 2014.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı	Can Razbonyalı
Doğum Yeri, Tarihi	Ankara, 14.12.1984
Uyruğu	T.C.
Telefon	(0534) 382 55 55
E – Posta	canrazbonyali@gmail.com

EĞİTİM

Doktora	Okan Ün., Bilgisayar Mühendisliği	2012-2016
Yüksek Lisans	Trakya Ün., Bilgisayar Mühendisliği (İTÜ'den ders transferi yapıldı)	2010-2011
Lisans	Maltepe Ün., Bilgisayar Mühendisliği	2003-2007
Lise	Özel Marmara Koleji, Matematik Bölümü	2002-2003
	Burak Bora Anadolu Lisesi	1998-2002

İŞ DENEYİMİ

Cybersoft	Ar – Ge	2015-
SFS Danışmanlık	Yazılım Geliştirme	2014-2015
Turkcell Teknoloji	Yazılım Geliştirme	2014-2014
Huawei Teknoloji	Yazılım Geliştirme	2012-2014
	Askerlik Görevi	2011-2012
BYM Epistem	Yazılım Geliştirme	2011-2011
Okan Üniversitesi	MYO, Bilgisayar Programı	2009-2011
İTÜ	Müh.Yön.(Lisans Üstü Ders Alındı)	2008-2009
Tradesoft	Yazılım Geliştirme	2007-2008

YAYINLARI

Razbonyalı, C., Bilgisayar Oyunları ve Spor Müsabakaları, AB2014 Akademik Bilişim Konferansları, 5-7 Şubat 2014, Mersin Üniversitesi, Mersin.

Razbonyalı, C., Uyar, A., Makine Öğrenmesi İle Ürün Sınıflandırma İncelemesi, AB2014 Akademik Bilişim Konferansları, 5-7 Şubat 2014, Mersin Üniversitesi, Mersin.

Razbonyalı, C., Güvenoğlu, E., Traditional Data Storage Methods and the Big Data Concepts, 2016 (Baskıda).