T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ORTAM VERİ TABANLARI VE UYGULAMALARI

Deniz TAŞKIN

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı
Danışman : Yrd. Doç Dr. Nurşen Suçsuz

Edirne 2004

T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ORTAM VERİ TABANLARI VE UYGULAMALARI

Deniz TAŞKIN

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Bu tez 03/08/2004 tarihinde aşağıdaki jüri tarafından kabul edilmiştir.

Yrd.Doç.Dr. Nurşen SUÇSUZ
Danışman

Yrd.Doç.Dr.Seyfettin DALGIÇ
Üye

Yrd.Doç.Dr. Erdem UÇAR
Üye

ÖZET

Yüksek Lisans Tezi, Çoklu Ortam Veri Tabanları Ve Uygulamaları, T.C. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Bilgi teknolojisindeki hızla artan gelişmeler sonucunda, toplanan, derlenen ve saklanan veri tiplerinde de değişiklikler olmuştur. Daha önceleri salt metin girdi ve çıktısı yapabilen bilgisayarlar, teknolojik imkanların ilerlemesiyle görüntü, ses, akan görüntü gibi çoklu ortam verilerini de işleyebilir hale gelmişlerdir. İnsan beyninde bir salt metinden daha fazla iz bırakan çoklu ortam verileri, bilgisayarlara olan ilgiyi daha da arttırmıştır. Bu eğilime bağlı olarak; görevleri bilgiyi derleme, toplama ve saklama olan veri tabanı yönetim sistemleri de çoklu ortam verileri için yetersiz kalmışlardır. Çoklu ortam verilerinin düzenlenmesi ve bu verilere büyük miktarda bilgi bulunduran saklama alanlarından erişilmesi gerekliliği, çoklu ortam veri tabanı sistemi kavramı ihtiyacını doğurmuştur.

Bu çalışmada; mevcut olan çoklu ortam veri tabanı sistemi incelenmektedir. Çoklu ortam veri tabanı sisteminin genel yapısı, uzaysal veri tipleri ve uzaysal erişim metotları hakkında bilgi verilmektedir. Algoritmalar gerçek veri seti üzerine uygulanmaktadır.

Bu tez 2004 yılında yapılmıştır ve 95 sayfadan oluşmaktadır.

ANAHTAR KELİMELER: Sabit ızgara, Izgara dosyası, Dörtlü ağaç, Z sıralaması, Hilbert eğrisi, Gray kodu, R Tree.

ABSTRACT

Graduate Thesis, Multimedia Databases And Applications, T.C. Trakya University, Graduate School Of Natural And Applied Sciences, Department Of Computer Engineering.

The rapid enhancements in information technology have changed the types of collected, compiled and stored data. The computers that can only manipulate pure textual input and output before that, can process multimedia datas like image, sound and video after technological improvements. Multimedia data that leave more marks in human brain than pure text increased interest for computers.

Depending on this trendency database management systems that have tasks like collecting, compiling and storing data are insufficient for multimedia data sets. Tuning multimedia data sets and necessity of accessing storage space that includes great data brings about requirement of multimedia database system concept.

In this work, existing multimedia database system is studied. The information about basic structure of multimedia database system, spatial data types and spatial access methods ara given. The algorithms ara applied on real data sets.

This work is done in 2004 and consists of 95 pages.

KEYWORDS: Fixed Grid, Grid File, Quad Tree, Z-order, Hilbert Curve, Gray Code, R Tree

TEŞEKKÜR

Bu çalışmanın hazırlanmasında bana yol gösteren, destek ve yardımlarını esirgemeyen danışman hocam Yrd. Doç Dr. Nurşen SUÇSUZ'a, çalışmalarımda bana yardımcı olan Yrd. Doç Dr. Cavit TEZCAN ve değerli hocalarıma, çalışma arkadaşlarıma ve aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

1. GİRİŞ

Günümüzde çoklu ortam teknolojisinin gelişmesine paralel olarak, sayısal biçimde saklanan çoklu ortam veri (ses, görüntü, akan görüntü) miktarında da büyük bir artış olmaktadır. Bu engellenemez artış, çoklu ortam verisinin etkin bir biçimde işlenebilmesi için bir veri organizasyonunun gerekliliğini doğurmuştur. Geleneksel veritabanı sistemleri çoklu ortam verilerini etkin bir biçimde yönetememektedir. Çoklu ortam verileri çok büyük boyutta ve karmaşık ilişkilere sahiptirler. Birçok çoklu ortam uygulaması etkin bir indeksleme gerektirmektedir.

Bu doğrultuda verilerin düzenlenmesi ve etkin bir biçimde işlenebilmesi için uzaysal veritabanı sistemi geliştirilmiştir.

Çoklu ortam veritabanı sistemi, uzaysal parametrelere sahip nesnelerin özniteliklerini bularak, bu nesnelerin disk üzerine yerleştirilmesi ve sorgulanmasından sorumludur. Çoklu ortam veri tabanı sistemi, uzaysal niteliklere sahip verileri, etkin bir biçimde yönetebilmek için kumanda merkezi görevi yapmaktadır.

Bu çalışma beş bölümden oluşmaktadır. İlk bölümde çoklu ortam veri tabanı sistemi ve yapısı anlatılmaktadır.

İkinci bölümde çoklu ortam veri tabanı sistemi fiziksel katmanı ve erişim metotları anlatılmakta, algoritmalar gerçek verilere uygulanmaktadır.

Üçünü bölümde özellik çıkarıcı katman ve özellik çıkarımında kullanılan yöntemler anlatılmaktadır.

Dördüncü bölümde çoklu ortam veri tabanı sisteminin son kullanıcı tarafından kullanılan birimi görsel sorgu ara yüzü anlatılmaktadır.

Son bölümde çalışmadan elde edilen sonuçlar yer almaktadır.

2. ÇOKLU ORTAM VERİ TABANI SİSTEMİ

Son zamanlarda, uzaysal verinin yönetilmesine artan bir ilgi vardır. Uzaysal veri, doğasından gelen çok boyutluluk özelliğine sahiptir. Uzaysal veriye dayalı uygulamalar arasında coğrafi bilgi sistemleri (GIS), bilgisayar destekli tasarım (CAD), çoklu ortam veri tabanları, bilimsel veritabanları bulunmaktadır. [Gaede, 1998]

Çoklu ortam veritabanı görsel, işitsel verinin yorumlanarak saklanması ve gerektiğinde en hızlı biçimde geri çağrılabilmesi için kullanılmaktadır. Çok boyutlu bir veri, k boyutlu bir veri uzayında, k adet özellik tarafından bir nokta ile ifade edilmektedir. Çoklu ortam veri tabanlarında, çoklu ortam nesneleri genellikle birden çok boyutla, özellik vektörlerine bağlanmışlardır ve veri tabanında sorgulama işlemleri bu özellik vektörleri işlenerek gerçekleştirilmektedir. Özellik vektörlerine örnek olarak renk yoğunluğu, şekil tanımlama, metin tanımlama... v.b. verilebilir.

Çoklu ortam veri tabanı yönetim sistemi, tüm bu işlemlerin kusursuz biçimde yerine getirilmesinden sorumludur. Çoklu ortam veri tabanı sistemleri, veri modelinde çoklu ortam verileri içermeli ve uygulamalarında uzaysal indeksleme ve uzaysal kesişim için etkin algoritmalar sağlayan uzaysal veri tiplerini desteklemelidir. Çoklu ortam veri tabanı sisteminin kullanılmasındaki amaçlar;

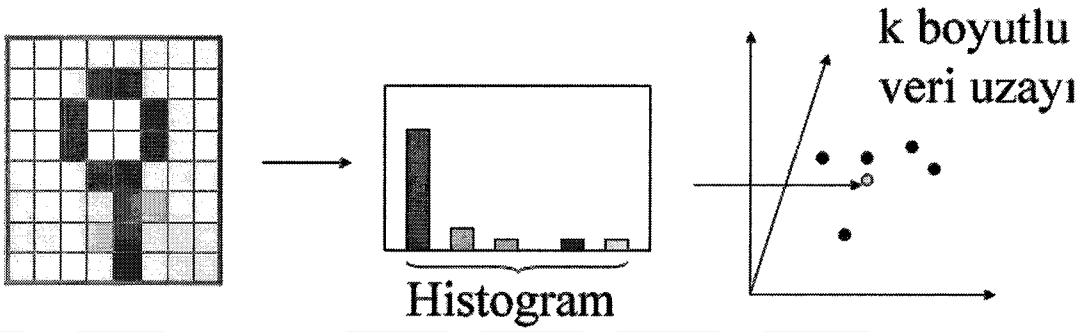
- Çoklu ortam öğelerinin bir veri tabanında saklanması
- Saklanan verilerin büyüklüğüne rağmen veri tabanında yüksek performanslı ekleme, arama ve silme algoritmalarının varlığı
- Çoklu ortam verileri saklanırken verinin belli özelliklere göre gruplandırılabilmesi
- Girilen çoklu ortam öğelerinin etkin bir biçimde içerdiği nesnel özelliklerine göre aranabilmesi.

Bir çoklu ortam veri tabanı sisteminin öngörülen katmanları sırasıyla şöyledir;

A- Fiziksel Katman: Fiziksel katman verinin disk üzerindeki konum bilgisini denetleyen katmandır. Verinin disk üzerinde tam olarak nerede tutulacağı bu katman tarafından belirlenir. Kullanılan veri tipi çoklu ortam verisi olduğundan dolayı konum bilgisinin belirlenmesinde görsel veya işitsel özellikler kullanılmaktadır. Bu özelliklerin konum bilgisinde kullanılmaları, klasik erişim metotları tarafından sağlanabilen bir

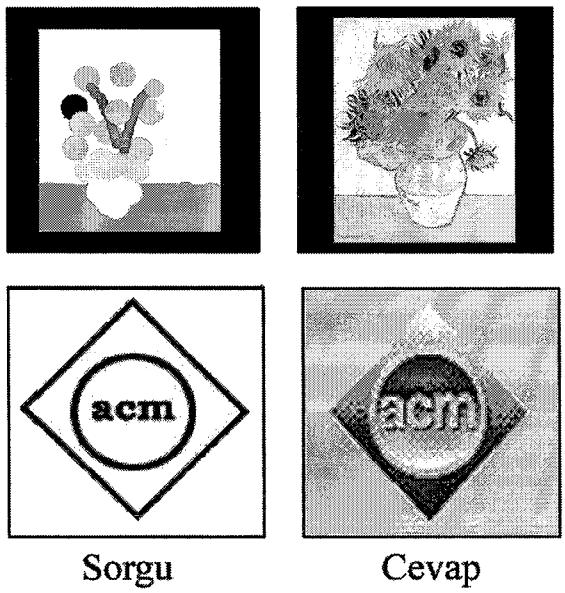
özellik değildir. Çoklu ortam verilerinin uzaysal özelliklerine göre değerlendirilmeleri uzaysal erişim metotları tarafından sağlanmaktadır.

B- Özellik Çıkarıcı: Çoklu ortam verilerinin sınıflandırılmasında kullanılacak, birbirlerine göre ayırt edici özelikleri belirleyen katmandır. Belirlenen bu veriler fiziksel katman tarafından kullanılmaktadır. Kullanılan özelliklere örnek olarak resimler için en küçük çevreleyen dikdörtgen ve renk yoğunluğu... v.b., ses dosyalarına örnek olarak sesin frekansı verilebilir.



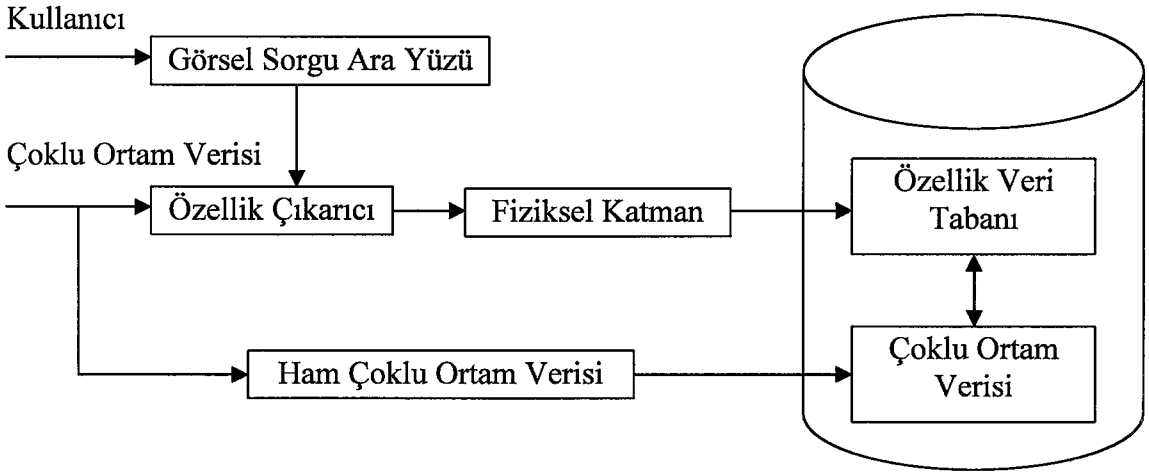
Şekil 2.1. Özellik çıkarıcı

C- Görsel Sorgu Ara Yüzü: Kullanıcıdan alınan uzaysal parametrelere bağlı olarak veri tabanını sorgulayıp geri dönen sonuçları kullanıcıya yansıtan birimdir. Bu katmanda grafik arabirim kullanılmaktadır. Grafik arabirim kullanıcıdan aldığı sorgu nesnesine karşılık gelen cevabı kullanıcıya iletmektedir.



Sekil 2.2. Görsel sorgu ara yüzü

D- Çoklu Ortam Verisi: Disk üzerinde çoklu ortam verilerine ait özellikler kaydedilmiş durumdadır. Buna ilave olarak bu özellik vektörleri ile bire bir ilişkili çoklu ortam verileri de disk üzerinde saklanmaktadırlar.



Şekil 2.3. Çoklu ortam veri tabanı sistemi

3. FİZİKSEL KATMAN

Uzaysal verilerin disk üzerinde tutulacakları konum bilgisi fiziksel katman tarafından belirlenir. Bu katman özellik çıkarıcı ile verinin saklandığı ortam arasındaki iletişimi sağlamaktadır. Özellik çıkarıcı tarafından iletilen özellik vektörünü kullanarak, verinin disk üzerindeki yerini belirlemektedir. Uzaysal veri tabanı sisteminde bulunan fiziksel katman, sistemin verimliliğiyle doğrudan alakalıdır. Bu katmanda kullanılan algoritmalar uzaysal erişim metotları olarak adlandırılır.

3.1. Uzaysal Erişim Metotları

Uzaysal erişim metotlarının tümünün ortak noktası uzaysal parametrelere sahip nesnelerin özelliklerini kullanarak, bu nesnelerin disk üzerine yerleştirilmesi ve sorgulanmasında etkinlik kazandırmasıdır. Klasik erişim metotlarından yığın, kuyruk gibi ana ve yardımcı bellek erişim metotları doğrusal özelliktedir. Bir verinin belleğin hangi noktasında bulunduğuna dair geliştirilmiş algoritmalar tek anahtar değer üzerinde çalışmaktadır. Bu yöntemler tek anahtar değere göre verileri disk izlerinde saklamaktadırlar.

Doğrusal erişim metotları çoklu ortam verilerini yönetmek için kullanılamazlar. Çoklu ortam verileri, özellik çıkarıcı katman tarafından sağlanan bir özellik vektörü tarafından ifade edilmektedir. Özellik vektörü *n* boyutlu bir veri uzayında verinin bir nokta ile temsil edilmesini sağlar. Uzaysal özelliklere sahip bu nesneler uzaysal erişim metotları ile yönetilmektedir.

Uzaysal erişim metotlarının etkinlikleri incelenirken yakın ve benzer uzaysal özelliklere sahip nesnelerin aynı yada birbirini takip eden bellek bölgelerinde olmaları göz önünde bulundurulmalıdır.

İki tip uzaysal erişim metodu bulunmaktadır;

1- *Uzay Kontrollü Yapılar:* Bu yapı iki boyutlu veri uzayını nesnelerin iki boyutlu düzlemdeki dağılımlarına bağlı olarak dikdörtgen hücrelere bölmektedir. Nesneler bazı geometrik kriterlere göre hücrelere bağlanmaktadırlar.

2- *Veri Kontrollü Yapılar:* Ortak özelliklere sahip bir grup nesneyi aynı bölgede tutarak nesneleri bölümleme yöntemini kullanan yapılardır.

3.1.1. Uzay kontrollü yapılar

Çok boyutlu verilerin indekslenmesinde uzay kontrollü yapılar, veri uzayını hücre adı verilen bölümlere ayırmaktadır. Veri uzayı, hücrelere bölümlenerek veriler yönetilmektedir.

Hücrelere bölme işlemi dinamik yada statik olarak gerçekleştirilebilir. Eğer hücrelere bölme işlemi veri girişi yapılmadan önce gerçekleştirilmişse bu statik bölümlemedir. Eğer hücrelere bölme işlemi veri girişi yapılmadan önce gerçekleştirilmemişse ve veri girişi esnasında gerçekleştiriliyorsa dinamik bölümlemedir.

Uzay kontrollü yapılardan Sabit Izgara yapısı, statik; Izgara dosyası, Dörtlü Ağaç, Z sıralaması, Hilbert Eğrisi, Gray Kodu yapıları ise dinamik bölümleme yöntemini kullanmaktadır. Izgara yapısında veri uzayını bölümleme, çeşitlerine göre iki tip yöntem vardır. Veri uzayını statik olarak hücrelere bölümleyen sabit ızgara ve veri uzayını dinamik olarak hücrelere bölümleyen ızgara dosyası.

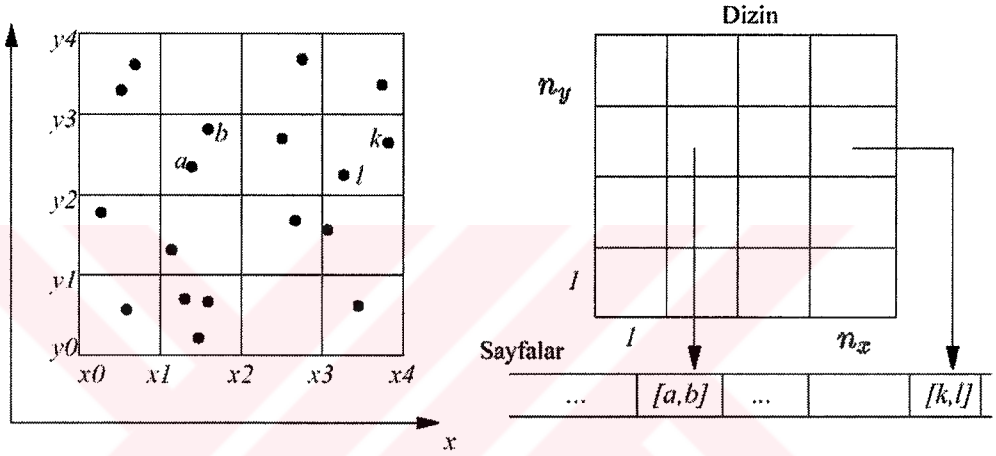
Hücrelere bölümleme işleminden sonra uzay kontrollü yapı yöntemlerine bağlı olarak bu hücreler ağaç, dizin gibi klasik erişim yöntemleriyle indekslenmektedirler. Uzay kontrollü yapılar, çoklu ortam verilerini kontrol etmek için veri uzayını klasik erişim yöntemleriyle indekslenebilecek bir hale dönüştürmektedirler. [Orenstein, 1988]

3.1.1.1. Sabit ızgara

Bu ızgara yapısına sahip, uzaysal erişim metodunda arama uzayı statik olarak dikdörtgen hücrelere bölünmektedir. Bölme sonucunda n_x x n_y sayısında hücrelerden oluşmuş bir ızgara yapısı meydana gelmektedir. Her bir c hücresi disk alanı ile ilişkilendirilmektedir. Bu ilişkilendirme işlemi iki boyutlu bir dizi kullanılarak doğrusal olarak geçekleştirilmektedir. Bir P noktası aynı hücre tarafından kapsanan diğer noktalarla birlikte disk alanında açılmış ortak bir sayfada sıralı bir biçimde

saklanmaktadır. Disk alanında nokta ile ilgili koordinat bilgilerine ilave olarak, diğer uzaysal özellikler de saklanmaktadır.

Şekil 3.1. daha önceden içine 18 adet nokta verisi eklenmiş sabit ızgara yapısının, dizin ve sayfa durumlarını göstermektedir. Veri uzayının başlangıç noktası (X_0,Y_0)'dır. Hücreleri disk alanı ile ilişkilendiren iki boyutlu indeks $DIR[1:n_x,1:n_y]$ dizisi tarafından tutulmaktadır. Dizinin her bir üyesi $DIR(i,j)$ c_{ij} hücresine bağlı noktaların bulunduğu disk sayfasının yerini göstermektedir. Eğer iki boyutlu arama uzayının boyutları $[S_x,S_y]$ ise her bir hücrenin dikdörtgen boyutları $[S_x/n_x,S_y/n_y]$'dir.



Şekil 3.1. Sabit ızgara

3.1.1.1.1. Sabit ızgara ekleme algoritması

Veri tabanına $P(a,b)$ noktası eklenmek istendiğinde sabit ızgara indeks yapısı bu noktanın disk üzerindeki yerini belirlemelidir. Öncelikle verilen a ve b koordinat değerlerinden $i=(a-x_0)/(S_x/n_x)+1$ ve $j=(b-y_0)/(S_y/n_y)+1$ değerleri hesaplanmaktadır. $DIR(i,j)$ dizin değerinden, kaydın hangi disk sayfasına uygun olduğuna karar verilmektedir. Disk sayfası belirlendikten sonra disk sayfasının kapasitesi kontrol edilmektedir. Kapasite uygun ise kayıt bu disk sayfasında sıralı bir şekilde saklanmaktadır. Kapasite dolu ise mevcut bir taşma alanı olup olmadığı kontrol edilmektedir. Eğer taşma alanı yoksa yeni bir taşma alanı oluşturarak gerekli düzenlemeler yapılmaktadır.

3.1.1.1.2. Sabit ızgara nokta sorgulama algoritması

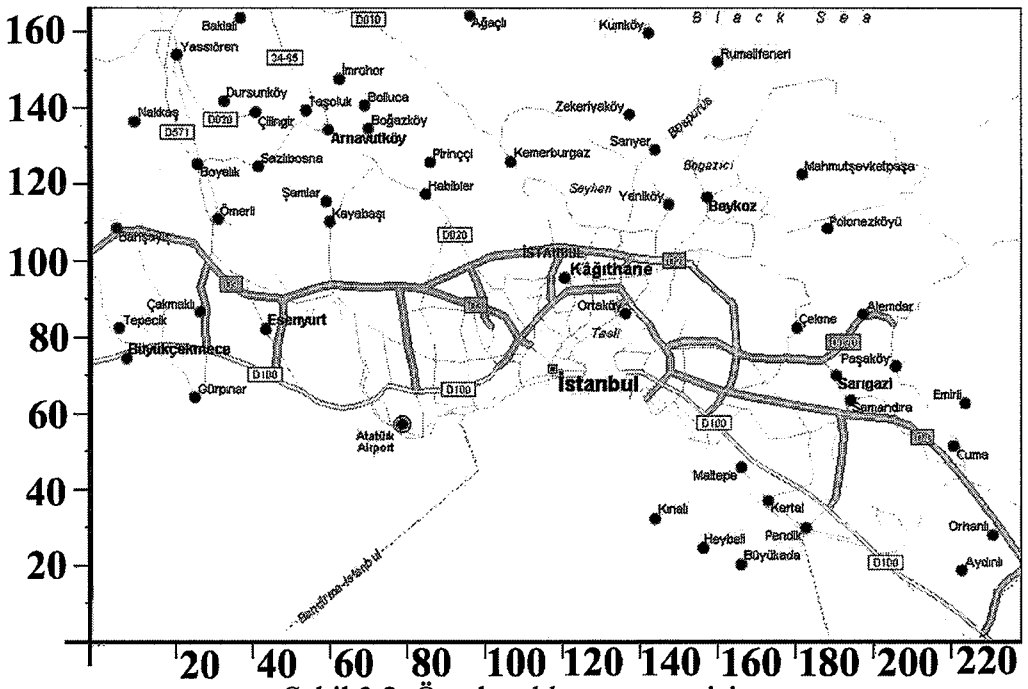
Veri tabanında $S(a,b)$ noktası aranırken, sabit ızgara indeks yapısı bu noktanın disk üzerindeki yerini belirlemeli, belirleyemiyorsa kaydı bulamadığını belirtmelidir. İlk olarak verilen a ve b koordinat değerlerinden $i=(a-x_0)/(S_x/n_x)+1$ ve $j=(b-y_0)/(S_y/n_y)+1$ değerleri hesaplanmaktadır. $DIR(i,j)$ dizin değerinden kaydın hangi disk sayfasında bulunabileceği belirlenmektedir. Disk sayfası belirlendikten sonra disk sayfasında sıralı olarak eklenmiş olan kayıtlar tek-tek kontrol edilmektedir. Eğer kayıt bulunursa kayıtla ilgili bilgiler disk sayfasından okunarak kullanıcıya iletilmektedir. Eğer kayıt bulunamamışsa ve taşma sahası bulunmuyorsa kullanıcıya "kayıt yok" hata mesajı dönmektedir. Taşma sahası bulunuyorsa, taşma sahasındaki kayıtlar sırayla aranmaktadır. Taşma sahasında kayıt bulunursa kayıtla ilgili bilgiler taşma sahasından okunarak kullanıcıya iletilmektedir. Aksi taktirde kullanıcıya "kayıt yok" hata mesajı döner.

3.1.1.1.3. Sabit ızgara aralık sorgulama algoritması

Veri tabanında sorgulanan $P1(x_1,y_1;x_2,y_2)$ penceresi ile üst üste olan yada aynı konuma sahip *c.dikdörtgen* alanlı c hücreleri kümesi S, sabit ızgara indeks yapısı tarafından belirlenmelidir. Öncelikle $i0=(x_1-x_0)/(S_x/n_x)$, $i1=(x_2-x_0)/(S_x/n_x)$, ve $j0=(y_1-y_0)/(S_y/n_y)+1$, $j1=(y_2-y_0)/(S_y/n_y)+1$ değeri hesaplanarak $DIR[i_0:i_1;j_0:j_1]$ dizin değerlerine karşılık gelen sayfalar S kümesinde tutulmaktadırlar. S kümesindeki her bir c_{ij} hücreleri ile ilişkili olan sayfalar ve içlerindeki kayıtlar, diskten okunarak kullanıcıya sonuç olarak dönmektedir.

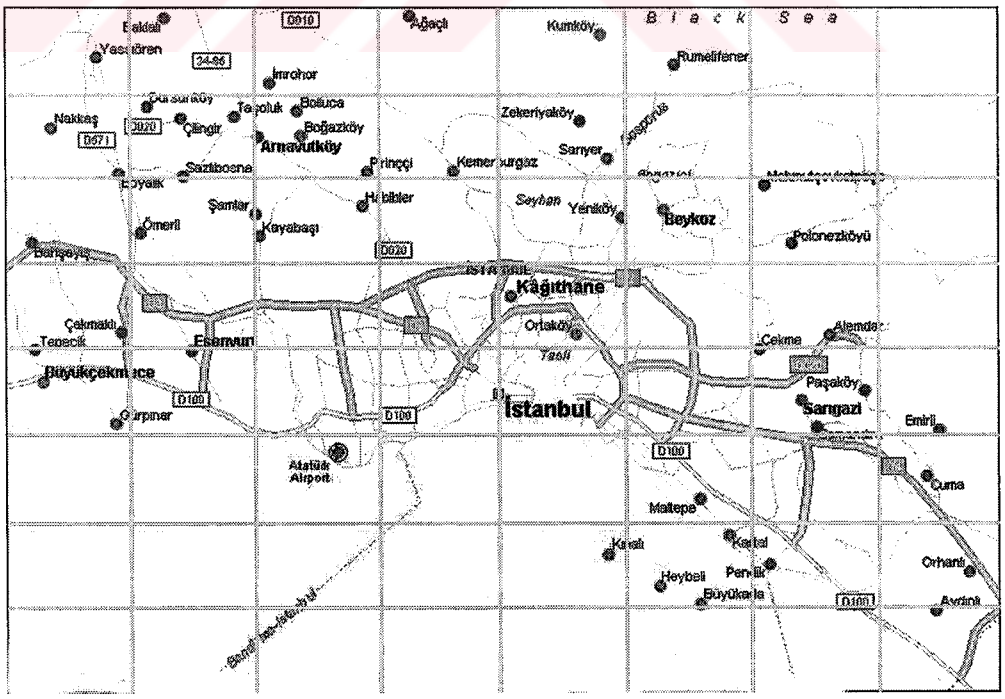
3.1.1.1.4. Örnek

Sabit ızgara sorgulama algoritmalarını uygulayabilmek için çoklu ortam verisi olarak İstanbul'un bazı semtleri, adaları, önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.



Şekil 3.2. Örnek çoklu ortam verisi

Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil etmektedir. 8X8'lik ızgara çözünürlüğünde resim 30cmX20cm'lik 64 adet hücreye bölünecektir. Arama uzayının başlangıç noktası $(x_0=0,y_0=0)$ ve sınırları (240,165)'dir. Her bir hücrenin kayıt kapasitesi ise 3 olarak belirlenmiştir.



Şekil 3.3. Sabit ızgara hücre yapısı

İki boyutlu nokta veriler, bulundukları konuma göre belli hücrelere dahil edileceklerdir. *i,j* indisine sahip bir hücrenin *P(x,y)* değerlerine sahip bir veriyi içinde bulundurabilmesi için $i=(x-x_0)/(S_x/n_x)+1$ ve $j=(y-y_0)/(S_y/n_y)+1$ koşullarını sağlaması gerekmektedir. Bu kurala göre veriler eklendikten sonra dizin yapısında bulunan kayıt sayıları aşağıdaki gibi olmaktadır.

1	1	1	1	1	1	0	0
2	4	4	1	2	0	0	0
1	1	3	0	1	1	2	0
1	0	0	0	2	0	1	0
3	1	0	0	0	0	4	1
0	0	1	0	0	1	0	1
0	0	0	0	1	3	1	1
0	0	0	0	0	0	0	1

Şekil 3.4. Sabit ızgara hücre yoğunlukları

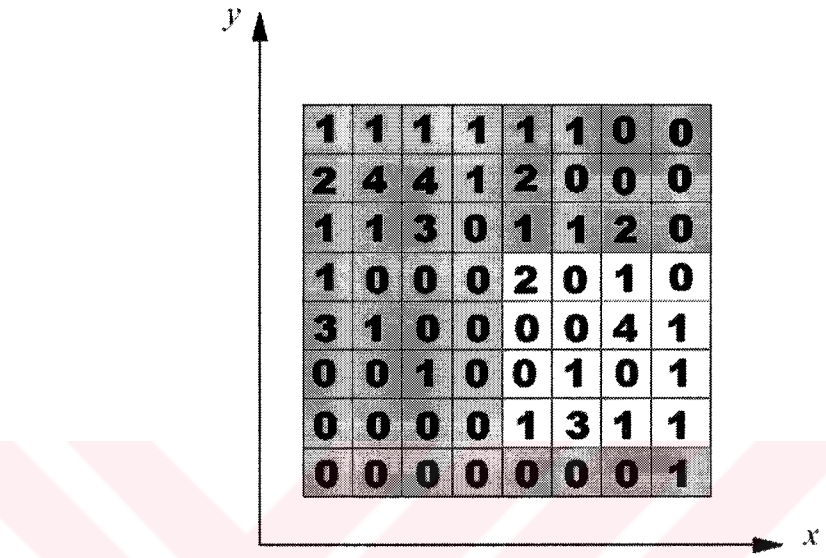
Sabit ızgara yapısında her bir bölge bir sayfa ile ilişkilidir. Yukarıdaki şekilde kayıt sayısı 3'ün üzerindeki sayfalarda taşma durumu oluşmuştur. Sayfalardaki kayıt sahaları dışında taşma sayfalarında da kayıtlar tutulmaktadır. Bu sayfada, kayıt sahalarından taşan kayıtlar bulunmaktadır.

3.1.1.1.5. Sonuç

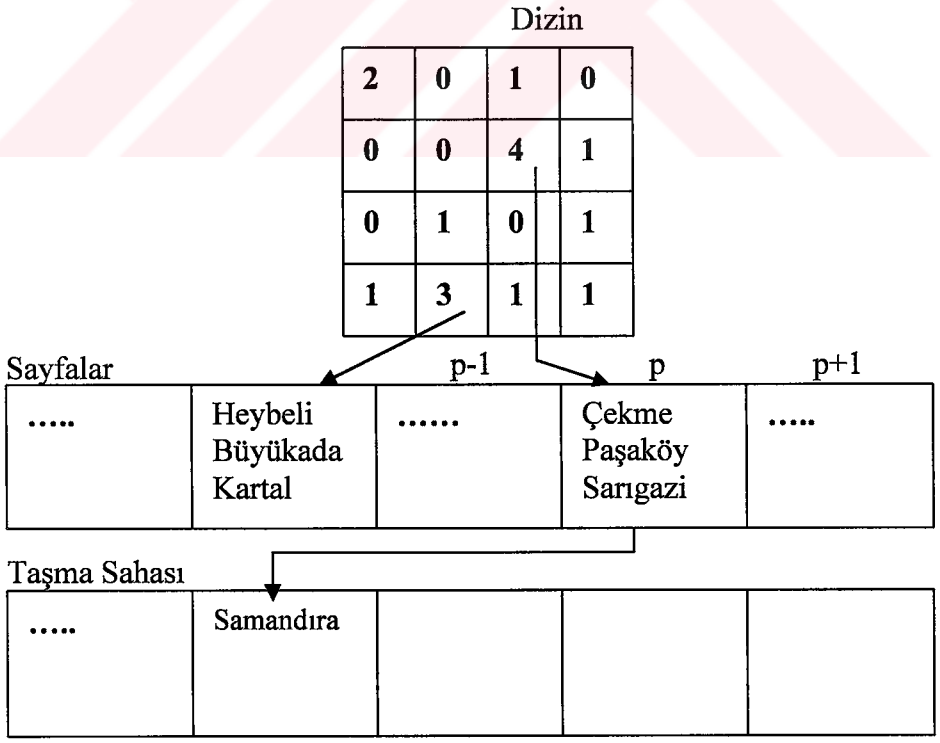
Sabit ızgara yapısında, nokta sorgulamanın etkinliği rahatça görülmektedir. Dizinin ana bellekte bulunduğu varsayıldığında, nokta sorgulama tek bir I/O işlemi gerektirmektedir. Pencere/Aralık sorgulama için gerekli I/O sayısı pencerenin kesiştiği hücre sayısıyla orantılıdır.

Izgara çözünürlüğü, indekslenecek olan *N* nokta sayısına bağlıdır. *M* adet nokta kapasitesine sahip bir sabit ızgara en az *N/M* adet hücre içermektedir. Her bir hücre ortalama olarak *M* adetten daha az nesne içermektedir. *M* adet noktadan daha fazla nokta ile ilişkilendirilmiş hücre, taşma meydana getirmektedir. Taşma sayfaları önceki sayfaya zincirlenir. Örnekte her bir sayfa en fazla 3 kayıt alabilmekte ve dağılım şekil 3.5'teki gibi olmaktadır. Çekme, Paşaköy, Sarıgazi, Samadıra noktalarının bulunduğu

hücre, 3'ten fazla nesne içerdiğinden taşma durumundadır. İlk eklenen 3 veri *p* sayfasında, taşan 4. kayıt ise *p* sayfasına bağlı taşma sayfasında saklanmaktadır. Bu yüzden nokta sorgulamadan taşma zincirinin *x*'inci sırasında bulunan bir nokta aranırken *x* adet I/O işlemi yapılması gerekmektedir.



a) İşleme giren hücreler



b) Hücrelerin disk üzerindeki yerleşimleri

Şekil 3.5. Sabit ızgara taşma sahası

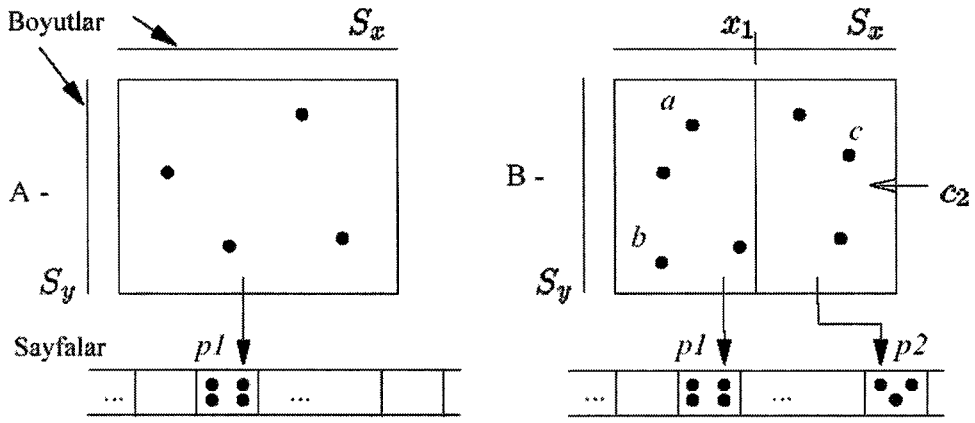
Eğer veriler arama uzayında rasgele dağılmışlar ise yukarıdaki olay çok seyrek gerçekleşmekte ve taşma zincirleri çok uzun olmamaktadır. Bunun gibi bir durum en iyi ihtimalle gerçekleşecektir. En kötü ihtimalle tüm noktalar tek bir hücrede bulunacak şekilde dağılabilirler. Bu durumda bir noktayı aramak doğrusal bir arama süresi gerektirmekte, buna ek olarak taşma zincirindeki arama süresi de bu süreyi arttırmaktadır. Bu gibi bir durum veritabanı boyutunun çok astronomik olarak büyümesiyle de ortaya çıkabilir. Nokta dağılımı düzenli olsa bile aynı sayfaya denk gelen yeni nokta ilaveleri büyük miktarlarda taşmaya yol açmaktadır. Bunun tam tersi de gerçekleşebilir. Yoğun silme işlemleri neredeyse boş sayfalar ortaya çıkarmaktadır.

Sonuç olarak bazı dağılımlar dışında sabit ızgara etkin bir indeksleme biçimi değildir.

3.1.1.2. Izgara Dosyası

Izgara dosyası yaklaşımında, sabit ızgara yaklaşımında olduğu gibi veri uzayı hücrelere bölünmüş ve her bir sayfa bir hücre ile ilişkilendirilmiştir. Farklı olarak ızgara dosyası yapısında bir hücre dolduğu taktirde taşma meydana gelmez ve kapasitesini aşan hücre iki ayrı alt hücreye bölünmektedir. Ardından ilk hücrenin boyutundan farklı, ilk hücreyle kapasiteleri aynı iki hücreye noktalar dağıtılmaktadır. Bölünmenin kayıt eklenirken gerçekleşmesi dinamik bölünme algoritması sayesindedir. Her bir c hücresi disk alanı ile ilişkilendirilmektedir. İlişkilendirme işlemi sabit ızgara yönteminde olduğu gibi iki boyutlu bir dizi kullanılarak doğrusal olarak gerçekleştirilir. Bir P noktası, aynı hücre tarafından kapsanan diğer noktalarla birlikte disk alanında sıralı bir biçimde saklanmaktadır. Disk alanında nokta ile ilgili koordinat bilgilerine ilave olarak diğer uzaysal özellikler de saklanmaktadır.

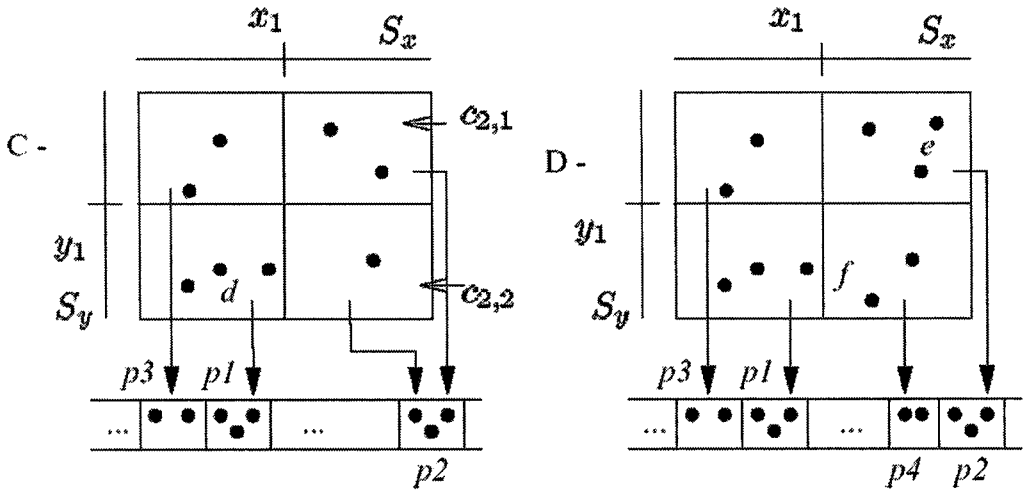
Hücreleri disk alanı ile ilişkilendiren iki boyutlu indeks $DIR[1{:}n_x, 1{:}n_y]$ dizisi tarafından tutulmaktadır. Yapısı sabit ızgara yöntemindeki ile aynıdır. Dizinin her bir üyesi $DIR(i,j)$, c_{ij} hücresine bağlı noktaların bulunduğu disk sayfasının yerini göstermektedir. Sabit ızgara yönteminden farklı olarak birbirini tamamlayan iki hücre aynı sayfaya bağlı olabilir. Şekil 3.6.'da hücreler ve sayfalar arasındaki ilişkiler gösterilmektedir.



Şekil 3.6. Izgara dosyası

Izgara dosyası yapısında, uzayı bölen hücreler sabit ızgara yönteminden farklı olarak dinamik olarak belirlenmektedirler. Veri uzayında kayıt kapasitesi doluncaya kadar tüm uzayı kapsayan tek bir hücre bulunmaktadır. Tüm veriler bu hücre içinde bulunurlar. Hücre, kapasitesini aştığında bölünme algoritması gereğince iki alt hücreye bölünmektedir. Hücrelerin hangi oranlarda bölünecekleri yine bölünme algoritması tarafından belirlenmektedir.

Ölçek S_x ve S_y değerleri, koordinat ekseni boyunca hücre bölümleri tanımlayan doğrusal dizilerdir. Ölçek dizinde bulunan her bir değer hücrelerin ayrıldığı koordinatları göstermektedir. Bu değerler ekleme, bölünme ve arama algoritmalarında kullanılmaktadır.



Şekil 3.7. Izgara dosyası bölünme işlemi

3.1.1.2.1. Izgara dosyası ekleme algoritması

Veri eklenmeye başlanmadan önce veri uzayında kayıt bulunmamaktadır. $S_x=[]$ ve $S_y=[]$ ölçek dizileri boştur ve veri uzayı hücrelere bölünmemiştir. Veri tabanı veri eklenmesinden önce uygun başlangıç durumuna getirilmelidir. Veri uzayı tek hücreden oluşan bir biçime getirilmektedir. Bunun için $S_x=[0]$ ve $S_y=[0]$ değerleri verilmektedir.

Veri tabanına $P(a,b)$ noktası eklenmek istendiğinde ızgara dosyası indeks yapısı bu noktanın disk üzerindeki yerini belirlemelidir. S_x, S_y ölçek değerlerinden $S_x=[i]<a$ ve $S_y=[j]<b$ koşuluna uyan en büyük (i,j) ikilisi seçilmektedir. P noktasını kapsayan hücre (i,j) indisine sahip olacaktır.

Hücre belirlendikten sonra hücrenin kapasitesi kontrol edilmektedir. Kapasite uygun ise kayıt bu hücre ile ifade edilen disk sayfasına kaydedilmektedir. Kapasite dolu olduğu taktirde bölünme algoritması gereğince hücre ikiye bölünecektir.

Bölünme gerçekleştikten sonra kayıtlar kendilerini kapsayan hücrenin disk sayfasına kaydedilmektedirler.

3.1.1.2.2. Izgara dosyası bölünme algoritması

Hücre ikiye bölüneceği zaman mümkün olan tüm bölünme tiplerinden en uygun olanı seçilmelidir. M kayıt kapasitesine sahip bir hücre $\frac{M+1}{2}$ ve $\frac{M-1}{2}$ adet kaydı içinde bulunduran iki hücreye bölünmelidir. Bölünme işlemi x ve y ekseninde olmak üzere birbirini izleyen iki yönde gerçekleşmektedir. İlk bölünme $\frac{M+1}{2}$ ve $\frac{M-1}{2}$ adet kaydı içinde bulunduracak şekilde x eksenine göre iki alt hücreye bölünmektedir. Bu bölünmeyi takip eden bir sonraki bölünme ise y eksenine göre olmaktadır.

İki hücre oluştuktan sonra, iki hücreyi birbirinden ayıran koordinat değeri, x ve y eksenine bağlı olan bölünme tipine göre S_x yada S_y dizisine atılmaktadır. Ölçek dizisi küçükten büyüğe doğru sıralanmaktadır.

3.1.1.2.3. Izgara dosyası nokta sorgulama algoritması

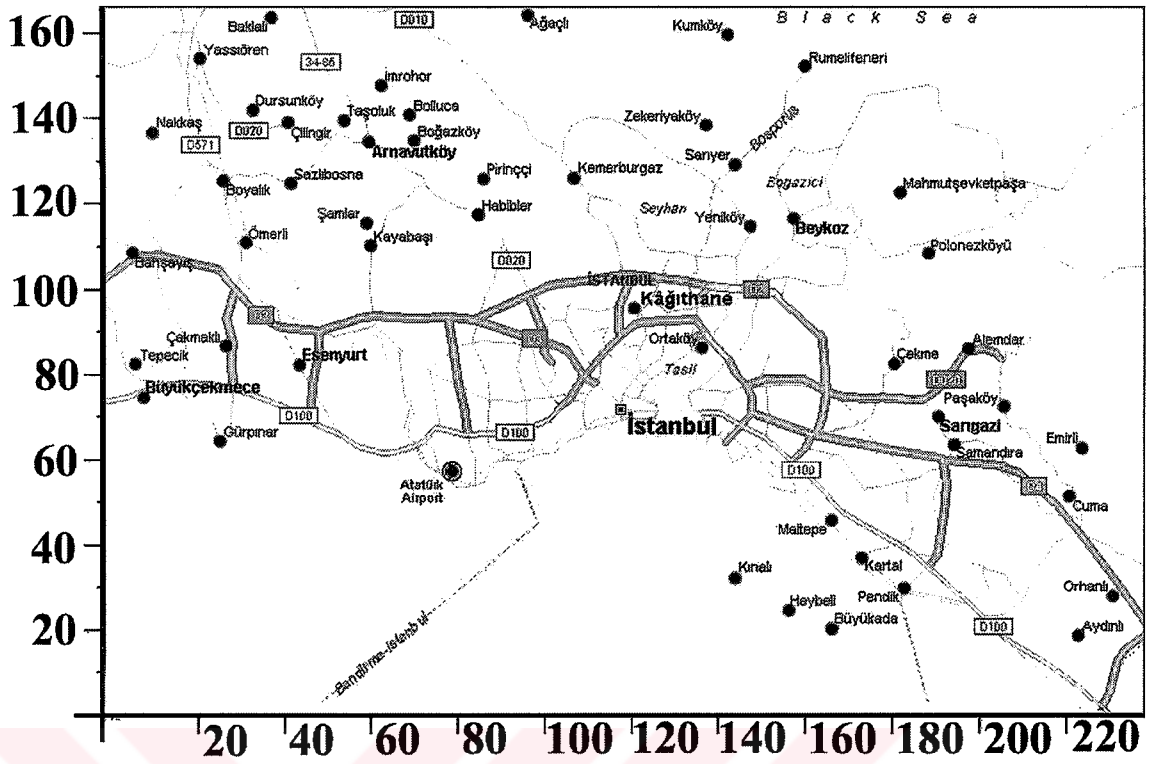
Veritabanında *S(a,b)* noktası aranırken ızgara dosyası indeks yapısı, bu noktanın disk üzerindeki yerini belirlemeli, belirleyemiyorsa kaydı bulamadığını belirtmelidir. Öncelikle verilen *a* ve *b* koordinat değerlerinden $S_x=[i]<a$ ve $S_y=[j]<b$ koşulunu sağlayan en büyük *i* ve *j* değerleri belirlenmektedir. *i, j* indis değerine sahip hücreye ait disk sayfasına *DIR(i,j)* indeks dizisi yoluyla ulaşılmaktadır. Disk sayfası belirlendikten sonra disk sayfasına sıralı olarak eklenmiş kayıtlar tek-tek kontrol edilmektedir. Eğer kontrol işlemi sırasında kayda rastlanırsa kayıtla ilgili bilgiler disk sayfasından okunarak kullanıcıya iletilmektedir. Eğer kayıt bulunamamışsa kullanıcıya "kayıt yok" hata mesajı döndürülmektedir.

3.1.1.2.4. Izgara dosyası aralık sorgulama algoritması

Veri tabanında $P(x_1,x_2:y_1,y_2)$ penceresi sorgulandığında ızgara dosyası indeks yapısı, bu pencerenin kapsadığı alan ile kesişen tüm nesnelerin bilgilerini geri döndürmelidir. Öncelikle pencere ile kesişen hücreler belirlenmektedir. Bunun için $S_x=[i_1]<x_1$, $S_x=[i_2]<x_2$, $S_y=[j_1]<y_1$ *ve* $S_y=[j_2]<y_2$ koşulunu sağlayan en büyük *i* ve *j* değerleri belirlenmektedir. Daha sonra $i_1:i_2,j_1:j_2$ indis değerleri arasında kalan hücrelerin *DIR(i,j)* indeks dizisi yoluyla bağlantılı olduğu disk sayfalarına ulaşılmaktadır. Belirlenen disk sayfalarında bulunan kayıt bilgileri kullanıcıya geri iletilmektedir.

3.1.1.2.5. Örnek

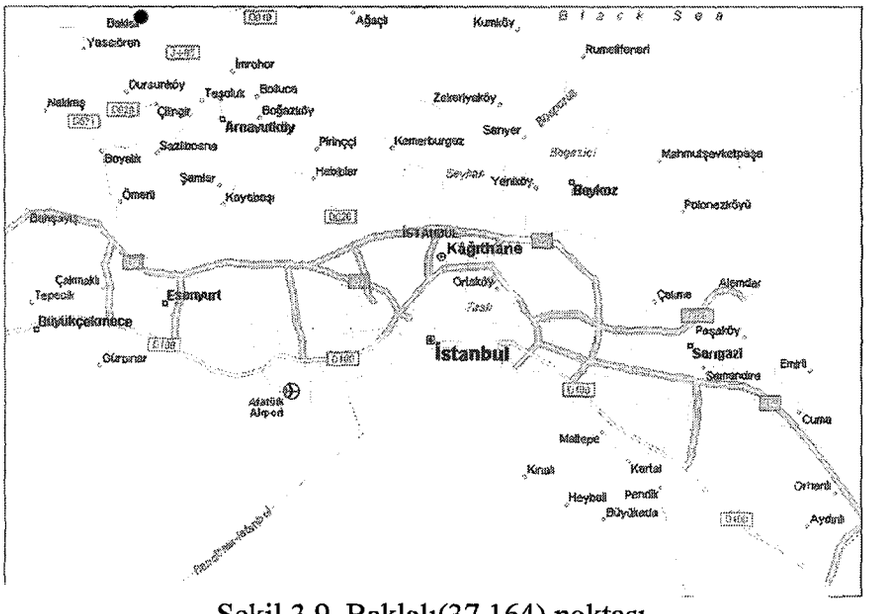
Izgara dosyası sorgulama algoritmalarını uygulayabilmek için çoklu ortam verisi olarak İstanbul'un bazı semtleri, adaları, önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.



Şekil 3.8. Örnek çoklu ortam verisi

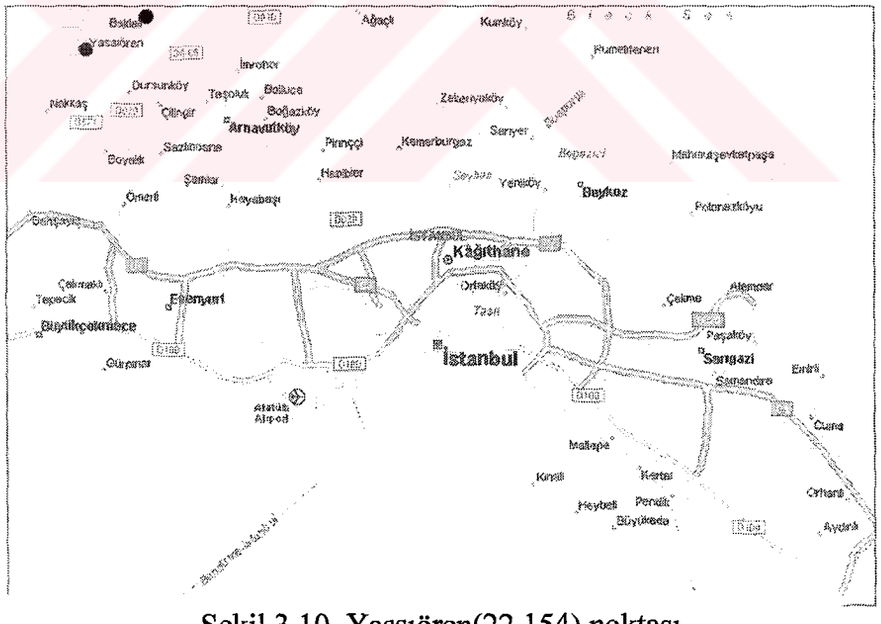
Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil etmektedir. Arama uzayının başlangıç noktası *(x_0=0, y_0=0)* ve sınırları (240, 165)'dir. Her bir hücrenin kayıt kapasitesi ise diğer yöntemlerle karşılaştırmada kolaylık sağlaması için üç olarak belirlenmektedir. Başlangıç durumu için *S_x=[0] ve S_y=[0]* dir.

1- Baklalı(37,164) noktası veri tabanına eklenmek istendiğinde; veri tabanında henüz herhangi bir kayıt bulunmamaktadır. Kayıt veri tabanının ilk sayfasına yerleştirilmektedir. *S_x=[0]* ve *S_y=[0]*'dır.



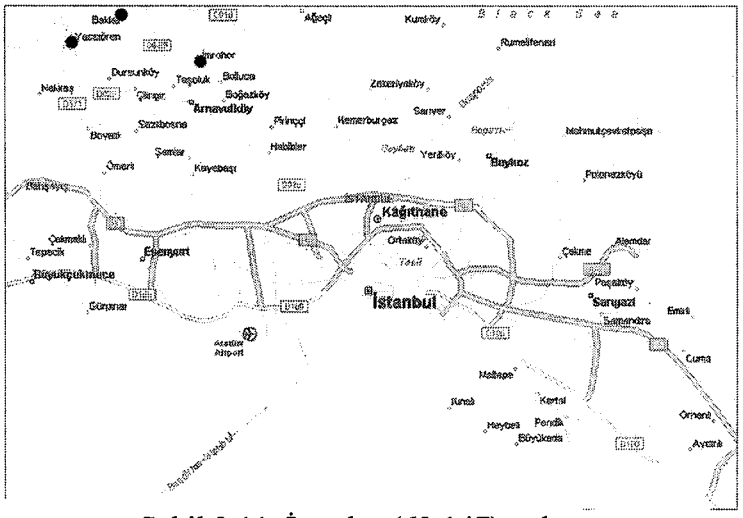
Şekil 3.9. Baklalı(37,164) noktası

2- Yassıören(22,154) noktası veri tabanına eklenmek istendiğine, veri tabanında 1 adet kayıt bulunmaktadır. Sayfa kapasitesi dolu olmadığı için, kayıt veri tabanındaki ilk sayfaya yerleştirilmektedir. $S_x=[0]$ ve $S_y=[0]$'dır



Şekil 3.10. Yassıören(22,154) noktası

3- İmrohor(63,147) noktası veri tabanına eklenmek istendiğine, veri tabanında 2 adet kayıt bulunmaktadır. Sayfa kapasitesi dolu olmadığı için, kayıt veri tabanındaki ilk sayfaya yerleştirilmektedir. $S_x=[0]$ ve $S_y=[0]$'dır



Şekil 3.11. İmrohor(63,147) noktası

4- Ağaçlı(98,164) noktası veri tabanına eklenmek istendiğine, veri tabanında 3 adet kayıt bulunmaktadır ve sayfa kapasitesi dolu olduğu için sayfada bölünme gerçekleşmektedir. Grid dosyası bölünme algoritmasına göre öncelikle *x* ekseninde bir bölünme gerçekleşir. 4 kayıt 3 farklı şekilde ikiye bölünebilmektedir.

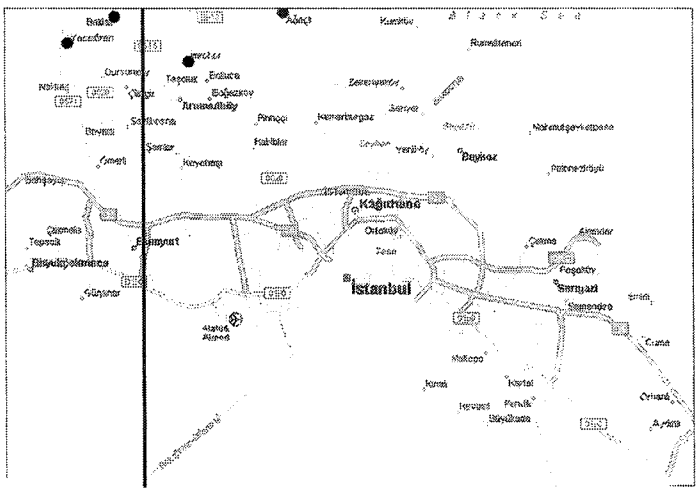
A- Baklalı, Yassıören – İmrahor, Ağaçlı

B- Baklalı, İmrahor – Yassıören, Ağaçlı

C- Yassıören, İmrahor – Baklalı, Ağaçlı

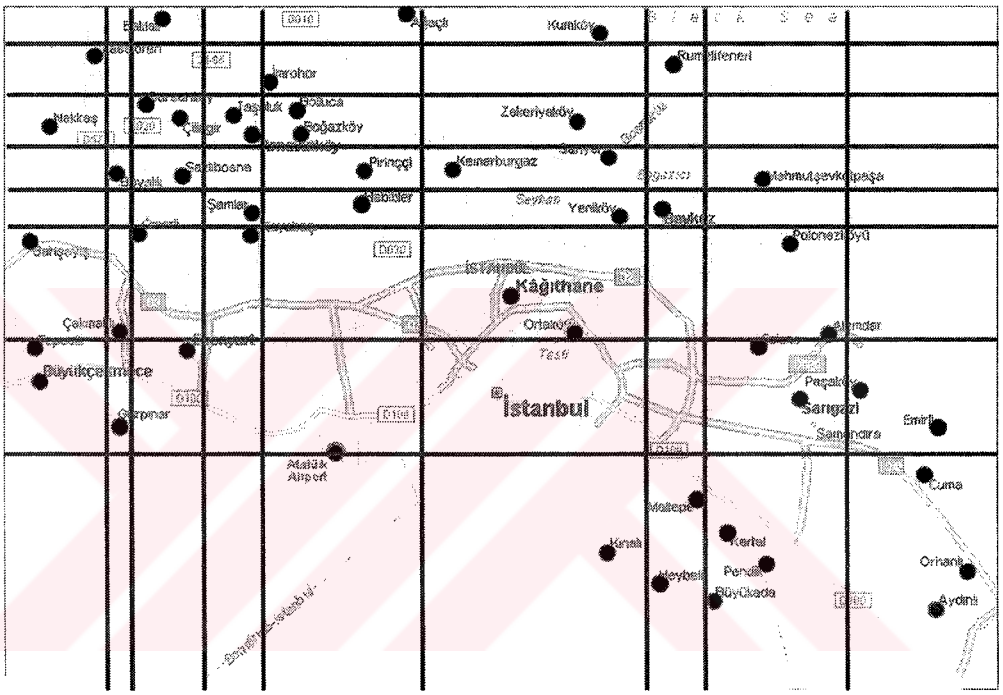
Bölünmelerden en uygunu Baklalı, Yassıören – İmrahor, Ağaçlı seçeneğidir. Bu seçenek doğrultusunda *x* ekseninde bölünme gerçekleşmektdir. Ölçek dizileri bu doğrultuda güncellenip küçükten büyüğe sıralanmaktadır. $S_x=[0,48]$ ve $S_y=[0]$'dır

Diğer kayıtların eklenmesi aynı şekilde devam etmektedir.



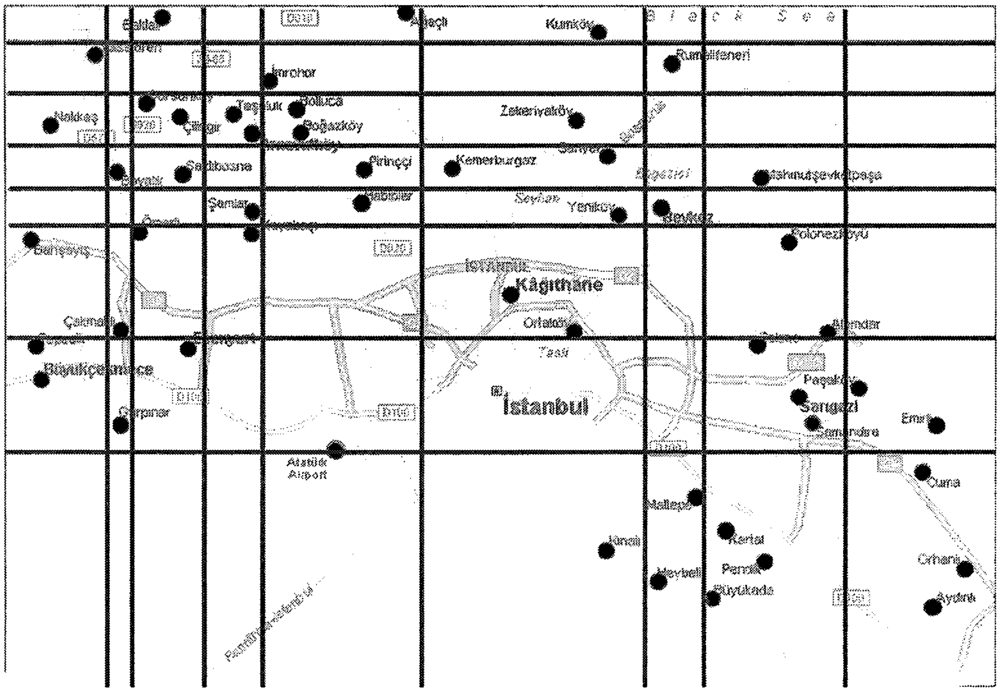
Şekil 3.12. Ağaçlı(98,164) noktası

50- Aydınlı(225,20) noktası veri tabanına eklenmek istendiğine, veri tabanında 49 adet kayıt ve 72 sayfa bulunmaktadır. Öncelikle arama algoritması kullanılarak Aydınlı (225,20) verisinin hangi sayfaya daha uygun olduğu belirlenmektedir. Arama algoritması, Aydınlı (225,20) verisinin koordinatlarını dikkate alarak, bu verinin yetmiş ikinci sayfaya ait olduğunu belirler. Sayfa kapasitesi dolu olmadığı için, kayıt veri tabanındaki yetmiş ikinci sayfaya yerleştirilir. S_x=*[0,25,31,48,62,102,155,170,204]* ve S_y=*[0,58,86,114,122,133,145,158]*'dir.



Şekil 3.13. Aydınlı(225,20) noktası

51- Samandıra(196,64) noktası veri tabanına eklenmek istendiğine, veri tabanında 50 adet kayıt ve 72 sayfa bulunmaktadır. Öncelikle arama algoritması kullanılarak Samandıra (196,64) verisinin hangi sayfaya daha uygun olduğu belirlenir. Arama algoritması Samandıra (196,64) verisinin koordinatlarını dikkate alarak, bu verinin altmış ikinci sayfaya ait olduğunu belirler. Sayfa kapasitesi dolu olmadığı için, kayıt veri tabanındaki altmış ikinci sayfaya yerleştirilir. S_x=*[0,25,31,48,62,102,155,170,204]* ve S_y=*[0,58,86,114,122,133,145,158]*'dir.



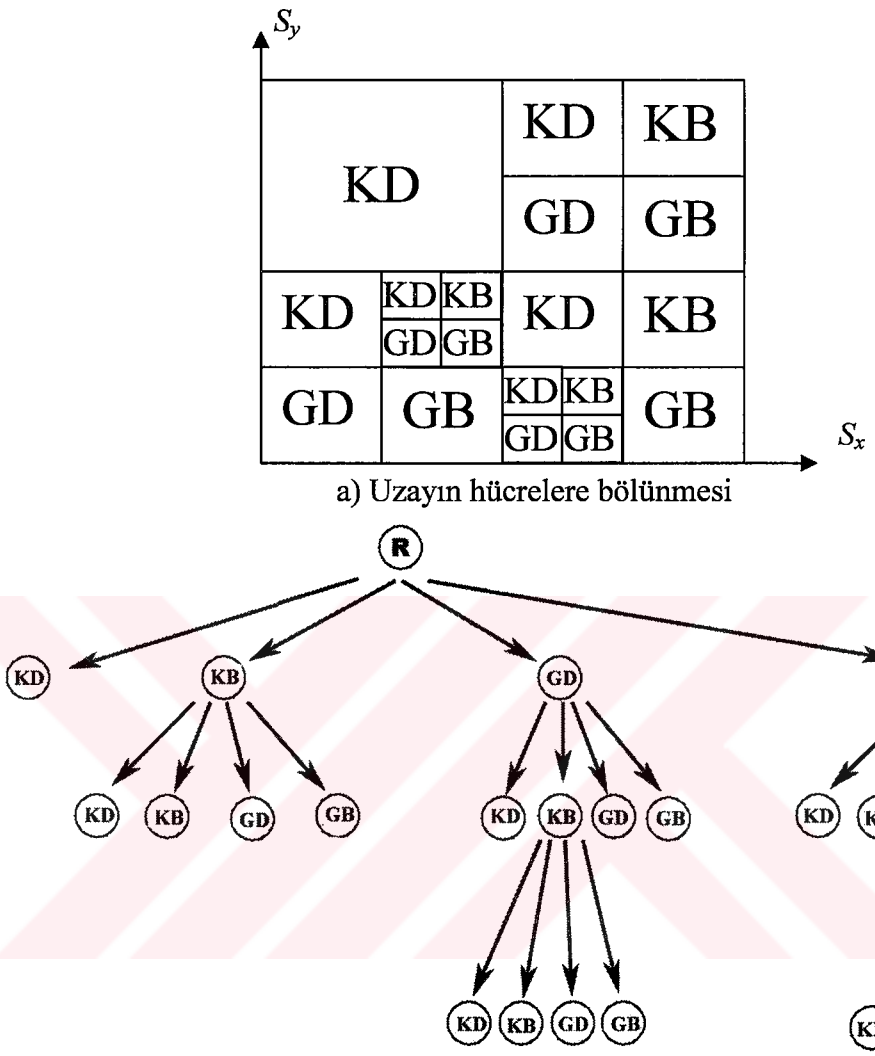
Şekil 3.14. Samandıra(196,64) noktası

3.1.1.3. Dörtlü Ağaç

Dörtlü ağaç yapısı, diğer uzay bölümlemeli yöntemlerden farklı olarak bölünen hücreleri dizin yapısından daha etkin olan ağaç yapısına dönüştüren bir yaklaşımdır. Bu yöntemin diğer bir farklılığı da bölünmeleri dinamik bir biçimde ve tek tip olarak gerçekleştirmesidir. Izgara dosyasında bölünmeler *x* ve *y* eksenlerine göre hücreleri ikiye bölen farklı iki tiptedir. Dörtlü ağaç yapısında ise hücreler eksenden bağımsız 4 eşit hücreye bölünmektedirler. Dörtlü ağaç yapısında bölünme özelliklerinin doğrultusunda her bir dalın 4 alt dalı bulunmaktadır. Yapraklar ise belli bir kapasiteye sahiptirler ve nesneleri tutarlar. Disk alanında nokta ile ilgili koordinat bilgilerine ilave olarak diğer uzaysal özellikler de saklanmaktadır. [Dyer,1982]

Dörtlü ağaç yapısında hücrelerin sayısını, bölünme koordinatlarını, sayfalarla olan ilişkilerini tutan bir dizi bulunmamaktadır. Bu tür bilgiler diğer yapılardan farklı olarak, ağaç yapısı ile belirlenmektedir. Arama uzayının sınırları S_x, S_y olduğunda dörtlü ağacın birinci bölünme seviyesinde arama uzayında, sabit $S_x/4$, $S_y/4$ boyutlarına sahip hücreler bulunmaktadır. Herhangi bir hücrenin alt hücrelere bölünmesi durumunda ağaç yapısı da ikinci seviyede bölünmektedir. Bu durumda alt hücrelerin

boyutları $S_x/16$, $S_y/16$ olur. Bu bilgiler yardımıyla verilen bir koordinat değerinin hangi hücre tarafından kapsandığı ağaç yapısından kolaylıkla belirlenmektedir.



a) Uzayın hücrelere bölünmesi

b) Ağaç gösterimi

Şekil 3.15. Dörtlü ağaç yapısı

3.1.1.3.1. Dörtlü ağaç bölünme algoritması

Dörtlü ağaç yapısında bölünme işlemi ızgara dosyasında olduğu gibi dinamik olarak gerçekleşmektedir. Arama uzayı her bir hücrede bulunan nesne sayısı, hücre kapasitesinin altına düşünceye kadar dört eşit parçaya bölünmektedir. Her bir çeyrek hücre Kuzeydoğu (KD), Kuzeybatı(KB), Güneydoğu(GD) ve Güneybatı(GB) olarak adlandırılmaktadır. İndeks yapısı her bir çeyreğin bir dal ile gösterildiği dörtlü ağaç yapısı ile ifade edilmektedir.

3.1.1.3.2. Dörtlü ağaç nokta sorgulama algoritması

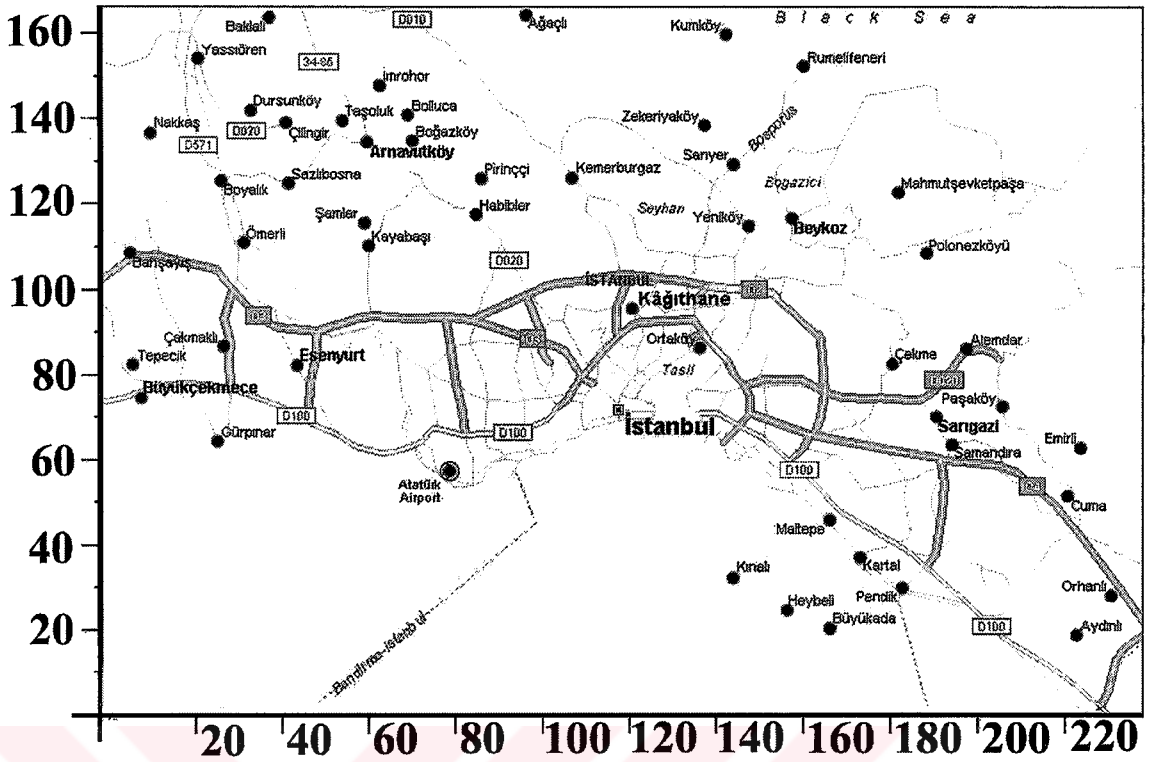
Veritabanında *S(a,b)* noktası arandığında dörtlü ağaç yapısı bu noktanın disk üzerindeki yerini belirlemeli, belirleyemiyorsa kaydı bulamadığını kullanıcıya belirtmelidir. Arama gerçekleştirilirken öncelikle verilen *a,b* koordinat değerlerinden *S(a,b)* noktasını kapsayan hücre belirlenmektedir. Her bir bölünme seviyesinde veri uzayı eşit parçalara bölündüğü için noktayı kapsayan hücrenin koordinatları ağaç yapısından hesaplanmaktadır. Arama işlemi yaprak düğüme ulaşılıncaya kadar tekrarlanır. Noktayı kapsayan yaprak hücreye erişildikten sonra bu hücreyle ilgili disk sayfasındaki sıralı biçimdeki kayıtlar tek-tek kontrol edilmektedir. Eğer kontrol işlemi sırasında kayda rastlanırsa, kayıtla ilgili bilgiler disk sayfasından okunarak kullanıcıya iletilmektedir. Eğer kayıt bulunamamışsa kullanıcıya "kayıt yok" hata mesajı döndürülmektedir.

3.1.1.3.3. Dörtlü ağaç aralık sorgulama algoritması

Veritabanında *P(x1,y1:x2,y2)* penceresi sorgulanırken, dörtlü ağaç yapısı bu pencerenin kapsadığı alan ile örtüşen tüm nesne bilgilerini geri döndürmelidir. Öncelikle pencere ile kesişen hücreler nokta sorgulamasında olduğu gibi belirlenirler. Her bir hücre ile bağlı olan disk sayfasına ulaşılmaktadır. Disk sayfasına sıralı olarak eklenmiş kayıtlar ve kayıt bilgileri kullanıcıya geri iletilmektedir.

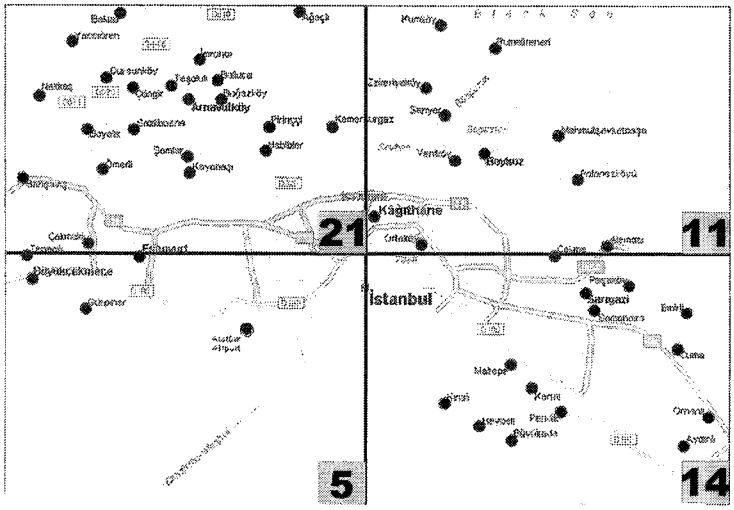
3.1.1.3.4. Örnek

Dörtlü ağaç sorgulama algoritmalarını uygulayabilmek için çoklu ortam verisi olarak İstanbul'un bazı semtleri, adaları, önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.

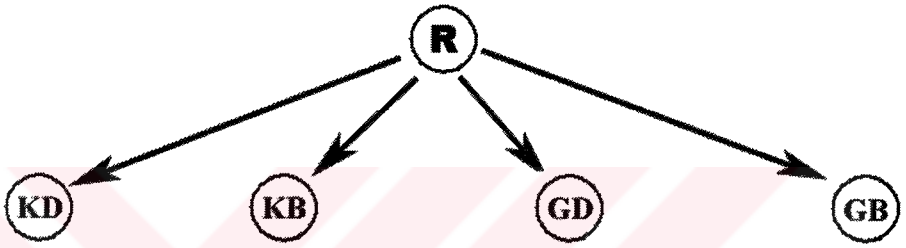


Şekil 3.16. Örnek çoklu ortam verisi

Diğer yöntemlerle karşılaştırılabilmesi açısından dörtlü ağaç yönteminde hücreler 3 kayıt kapasitesine sahiptir. Tüm veri seti 51 kayıt boyunda olduğu için tüm kayıtlar tek bir hücre ile ifade edilemezler. Bu yüzden bölünme algoritması gereğince veri uzayı dört eş hücreye bölünmektedir. Bölünen veri uzayı, her bir bölgede bulunan kayıt sayısı ve ağaç yapısı aşağıdaki gibi olmaktadır.



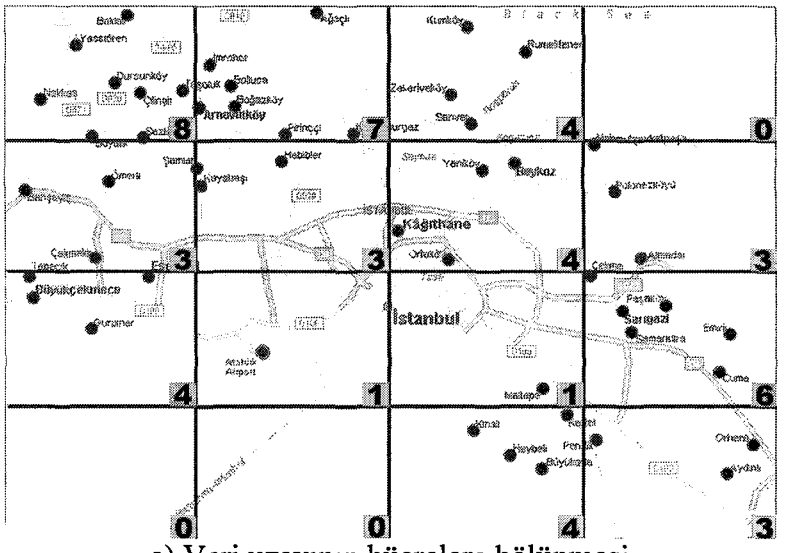
a) Veri uzayının hücrelere bölünmesi



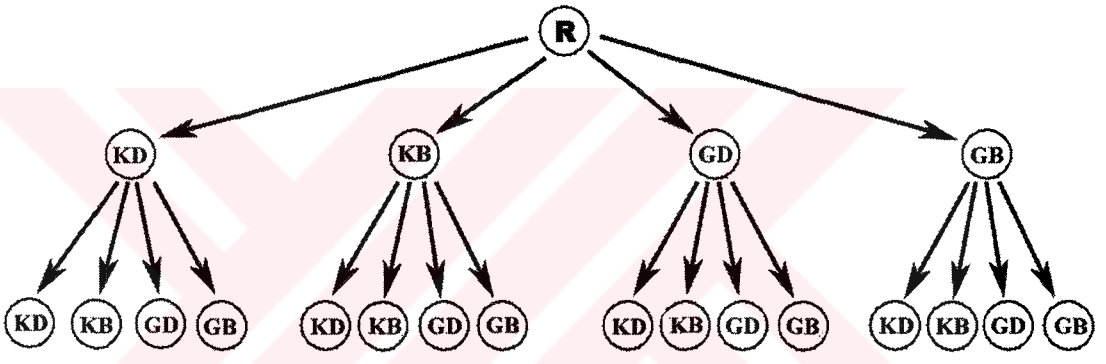
b) Ağaç gösterimi

Şekil 3.17. Dörtlü ağaç birinci seviye bölünme

Oluşan 4 hücrenin kapsadığı kayıt sayıları kapasitelerinin çok üstünde olduğundan dolayı her bir hücre tekrar eş dört alt hücreye bölünür. İkinci kademede gerçekleşen bölünme işleminden sonra veri uzayı, her bir bölgede bulunan kayıt sayısı ve ağaç yapısı aşağıdaki gibi olmaktadır.



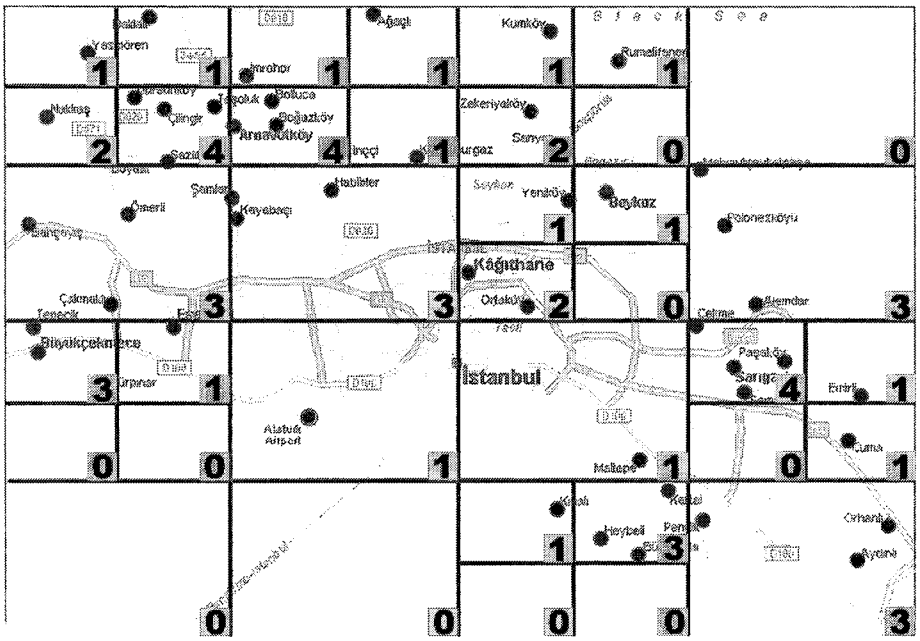
a) Veri uzayının hücrelere bölünmesi



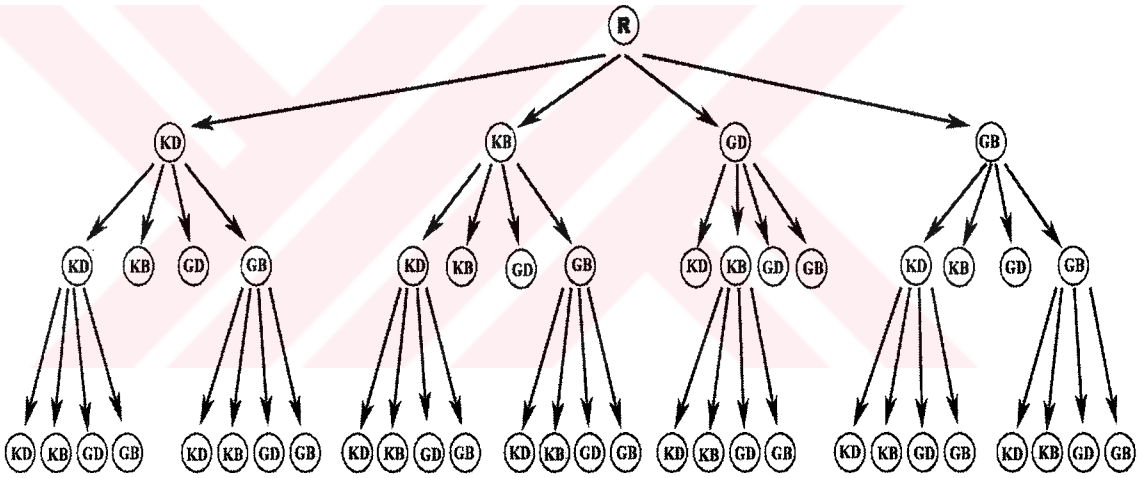
b) Ağaç gösterimi

Şekil 3.18. Dörtlü ağaç ikinci seviye bölünme

16 adet alt hücrenin kayıt sayıları incelendiğinde; *KD-KD, KD-KB, KB-KB, KB-GD, GD-KB, GB-KB* ve *GB-GD* hücrelerinin kapasitelerinin dolu olduğu görülmektedir. Kapasiteleri dolu olan hücreler algoritmaya göre alt hücrelere bölünürler. Üçüncü kademede gerçekleşen bu bölünme işleminden sonra veri uzayı, her bir bölgede bulunan kayıt sayısı ve ağaç yapısı aşağıdaki gibi olmaktadır.



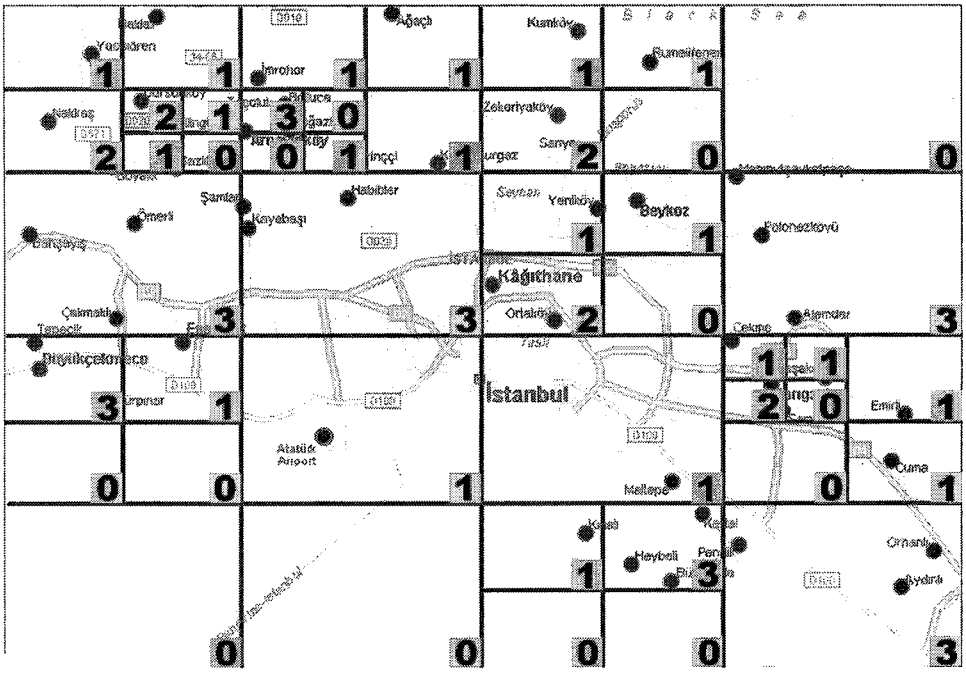
a) Veri uzayının hücrelere bölünmesi



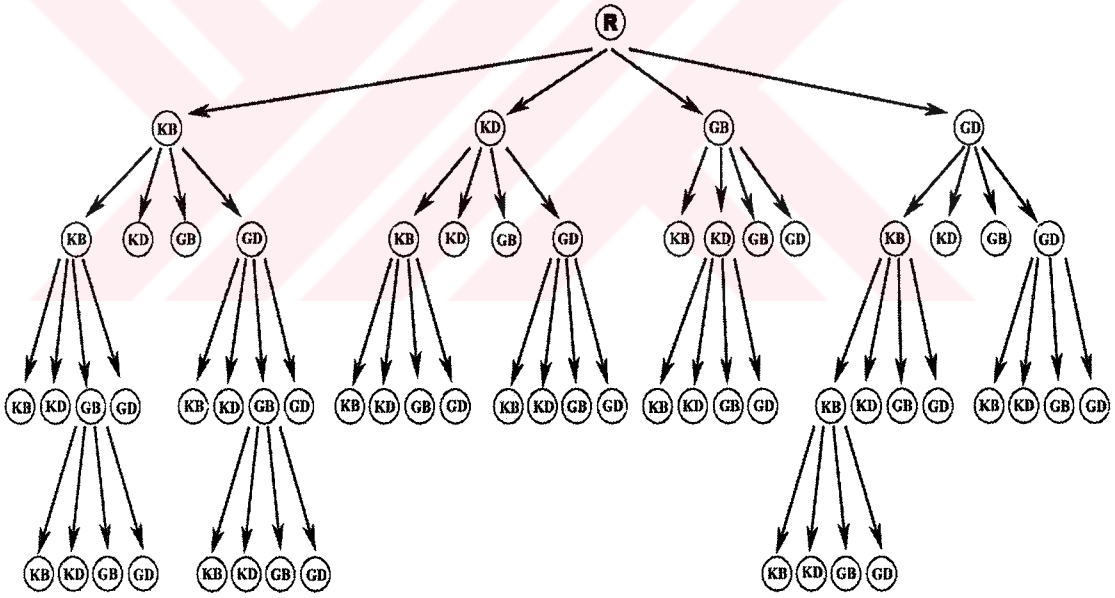
b) Ağaç gösterimi

Şekil 3.19. Dörtlü ağaç üçüncü seviye bölünme

Son kademede, kayıt kapasitelerini aşan alt hücre sayısı 3'tür. Bu hücreler de bölünme algoritmaları doğrultusunda bölünürler. Bu bölünme işleminden sonra veri uzayı, her bir bölgede bulunan kayıt sayısı ve ağaç yapısı aşağıdaki gibi olmaktadır.



a) Veri uzayının hücrelere bölünmesi



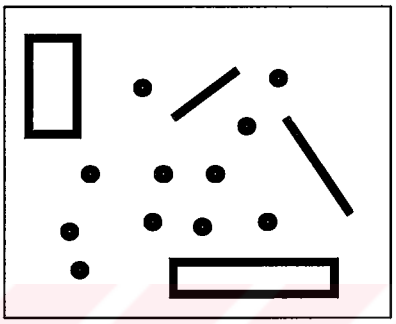
b) Ağaç gösterimi

Şekil 3.20. Dörtlü ağaç dördüncü seviye bölünme

En alt düzeydekiler dahil tüm hücreler ve kayıt sayıları kontrol edildiğinde herhangi bir kapasite aşımı söz konusu olmadığında tekrarlı olarak yürütülen bölünme algoritması son bulur.

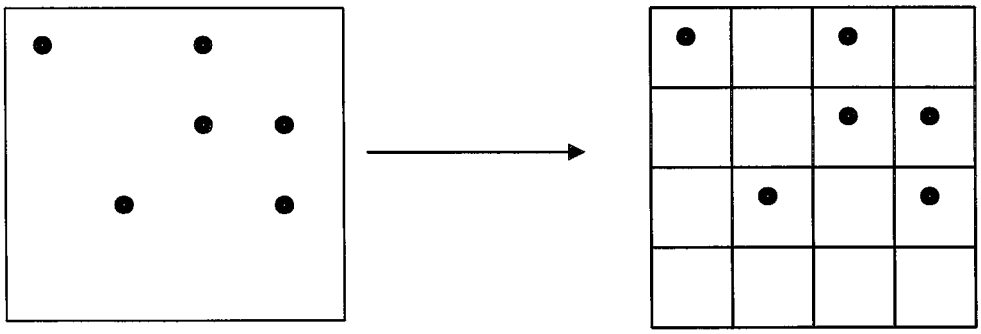
3.1.1.4. Z Sıralaması

Z sıralaması uzay kontrollü yapılardan, gelişmiş ve diğer yöntemlere nazaran daha etkin bir yaklaşımdır. Genel problem olan çok boyutlu uzaysal verinin tek boyuta indirgenmesine farklı bir çözüm getirmektedir. 2 boyutlu vektörsel bir veri disk üzerine her iki boyut büyüklükleri dikkate alınarak yerleştirilmelidir. Tek boyuta indirgenen veri, basit erişim metotlarından birisi (sıralı arama, B+ tree) seçilerek yönetilebilir.



Şekil 3.21. İki boyutlu veri uzayı ve uzaysal nesneler

n boyutlu verinin tek boyuta indirgenmesinde birçok yöntem geliştirilebilir. Z sıralaması yöntemi veri uzayını belirli bir ızgara çözünürlüğüne göre statik olarak hücrelere bölmektedir. Bölünen hücreler numaralandırılarak, numara sırasına göre disk üzerinde saklanmaktadır.

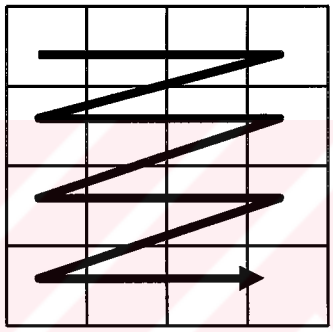


Şekil 3.22. Veri uzayının ızgaralara bölünmesi

Hücrelerin numaralandırılmasında dikkat edilmesi gereken nokta, yakın uzaysal özelliklere sahip verilerin disk üzerine yakın bölgelerde saklanması zorunluluğudur. Bu, özellikle aralık, en yakın komşu ve uzaysal birleşik tipindeki sorgulamalarda daha da önem kazanmaktadır. Bir yöntemin etkinliği karmaşık sorgulara verdiği yanıt süresi ile

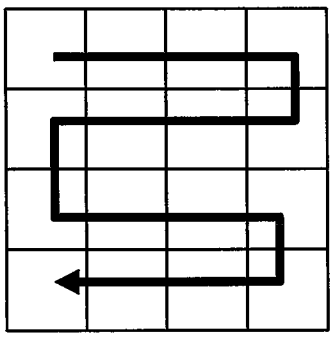
ölçülmektedir. Yakın uzaysal özelliklere sahip nesnelerin uzak disk bölümlerinde bulunması bu süreyi arttıracaktır.

Hücrelerin satır öncelikli olarak disk üzerine saklanmaları, soruna çok basit bir şekilde yaklaşım örneğidir. Bu tür bir çözüm bazı sorgulamalarda problem doğurmazken etkinliği tartışılabilir. Benzer x büyüklüğüne sahip nesneler yakın disk bölgesinde saklandığından, benzer y büyüklüğüne sahip nesneler birbirine uzak disk sayfalarında yer almaktadır. Bu çok boyutlu indeksleme mantığına haykırı bir durumdur. Dikkat edilirse bu tarz bir sıralama x ekseni göz önüne alınarak oluşturulmuş tek boyutlu bir indeks yapısıdır. y eksenine dayanan aralık sorgulamalar için uygun değildir.



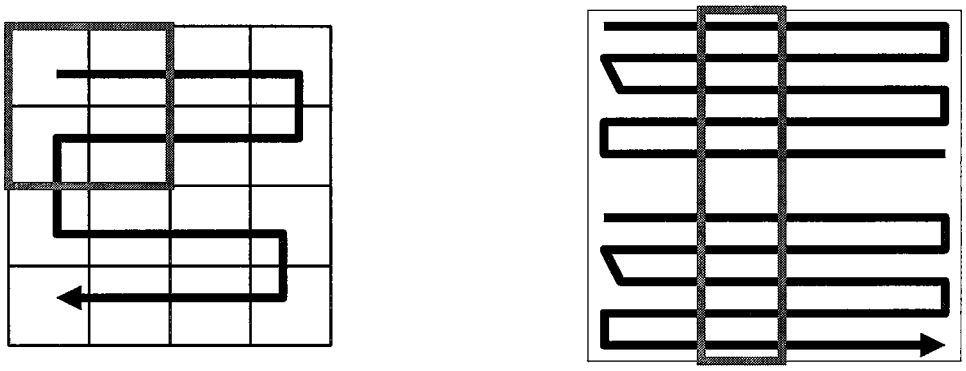
Şekil 3.23. Satır öncelikli sıralama

Hücreleri "S" harfi şeklinde sıralayarak disk üzerinde saklamak, satır öncelikli saklamaya bir alternatif olarak gösterilebilir. Bu durumda yakın uzaysal özelliklere sahip nesneler daha yakın disk bölgelerinde bulunurlar. Bu yöntemi de detaylı bir şekilde inceleyecek olursak ızgara çözünürlüğünün çok artması durumunda etkinliğinin çok azaldığı görülmektedir.



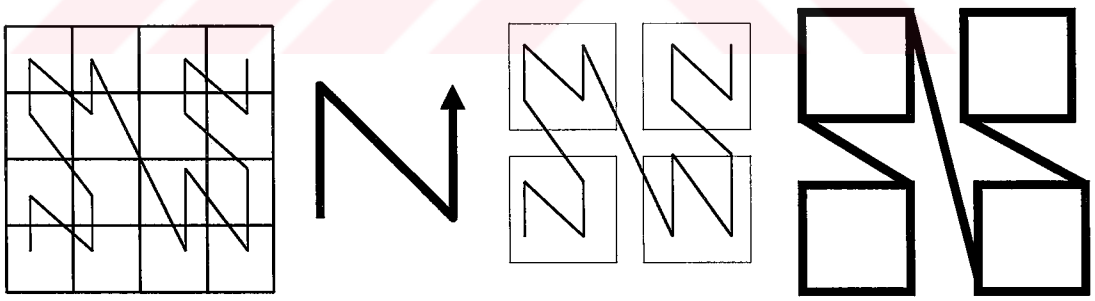
Şekil 3.24. "S" şeklinde sıralama

İki yöntemde de görüldüğü gibi ızgara çözünürlüğünden bağımsız olarak yakın özelliklere sahip nesneleri her konuda tutarlı bir şekilde yakın disk bölgelerine yerleştirecek etkin bir yöntem gerekmektedir.



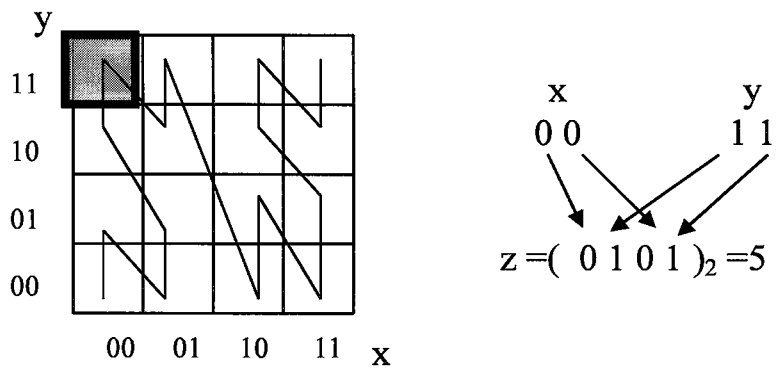
Şekil 3.25. "S" şeklinde sıralama

Z sıralaması bu problemleri ortadan kaldırmak üzere geliştirilmiş bir yöntemdir. Hücreler birbirini takip eden Z değerleri ile sıralandığında *N* harfine benzer bir şekil ortaya çıkar. Her 4 hücre *N* harfi şeklinde sıralanıp bir blok oluştururlar. Bloklar 4'lü biçimde yine *N* harfi oluşturacak şekilde sıralanırlar. Bu tüm hücreler sıraya girene dek yinelenen bir şekilde devam eder.



Şekil 3.26. Z sıralaması

Bu sıralamayı $z=f(i,j)$ şeklinde ifade etmek mümkündür. Herhangi bir hücrenin Z değerini hesaplamak için i ve j hücre indisleri kullanılmaktadır. Bit karıştırma yöntemi kullanılarak i,j indislerine sahip hücrenin Z değeri hesaplanmaktadır.



Şekil 3.27. Z değerinin hesaplanması

Elde edilen bu Z değerlerinin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın Z sıralaması değerlerine göre veriler disk üzerine sırayla yerleştirilebilirler. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır. Elde edilen Z değeri birincil anahtar olarak kullanılır ve verinin disk üzerine yerleştirilmesi B+ ağaç metoduyla gerçekleştirilir.[Rong, 1991]

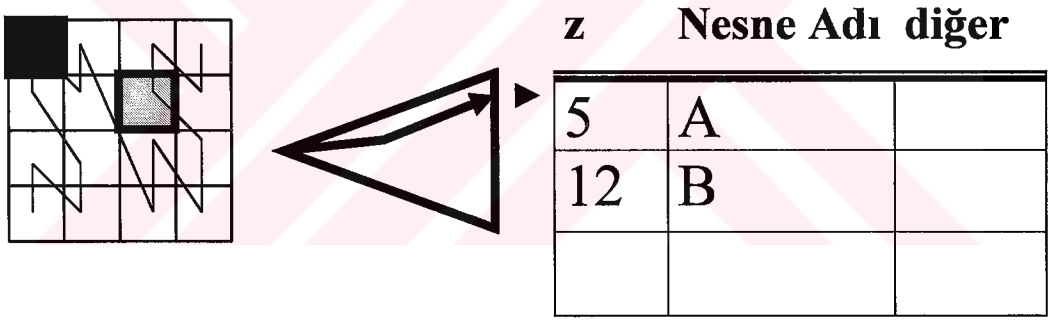
Z sıralaması değerinin disk üzerine yerleştirilmesinde B+ ağaç yönteminin kullanılmasında en önemli etmen bu metodun çoğu yaygın sistemlerde hali hazırda bulunması, hata bakımından temiz olmasıdır.

3.1.1.4.1. Z sıralaması ekleme algoritması

n x m ızgara çözünürlüğüne *Sn,Sm* sınır değerlerine sahip bir veri uzayında *S(x,y)* koordinatlarına sahip bir nokta eklenmek istendiğinde Z sıralaması ekleme algoritması kullanılmaktadır. Öncelikle verilen *x, y* koordinatlarından noktanın dahil olacağı hücrenin indis değeri hesaplanır. $x \le \frac{Sn}{n} i$ ve $x \le \frac{Sm}{m} j$ koşulunu sağlayan *i,j* değerlerden en küçüğü hücrenin indisi olarak belirlenmektedir. *z=f(i,j)* fonksiyonu kullanılarak indis değeri verilen hücrenin Z değeri hesaplanmaktadır. Belirlenen Z değeri B+ ağaç yapısı tarafından birincil anahtar olarak kullanılmaktadır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlenmektedir.

3.1.1.4.2. Z sıralaması nokta arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında, *S(x,y)* koordinatlarına sahip bir nokta aranırken Z sıralaması nokta arama algoritması kullanılmaktadır. Bu algoritma verilen x, y koordinatlarına göre nesnenin disk üzerindeki yerini belirlemekte ve sonuç olarak nesneye ait özellikleri kullanıcıya döndürmektedir. Öncelikle verilen *x,y* koordinatlarından noktanın içinde bulunabileceği olası hücrenin indis değeri hesaplanır. $(i=0,1,2,...n)$, $(j=0,1,2,...m)$ ve $x \le \frac{Sn}{n} i$, $x \le \frac{Sm}{m} j$ koşulunu sağlayan en küçük *i,j* değerleri hücrenin indisi olarak belirlenir. *z=f(i,j)* fonksiyonu kullanılarak indisi verilen hücrenin Z değeri hesaplanmaktadır. Belirlenen Z değeri B+ ağaç yapısı tarafından kullanılarak disk üzerinde nesne aranmaktadır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlendiğinde nesne ve özellikleri kullanıcıya bildirilir. Aksi taktirde "kayıt yok" hata mesajı verilir.

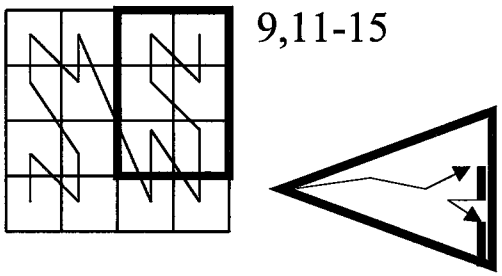


Şekil 3.28. Çoklu ortam verilerinin diskte tutuluş biçimi

3.1.1.4.3. Z sıralaması aralık arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında $P(x_1,y_1;x_2,y_2)$ penceresi tarafından kapsanana nesneler aranırken Z sıralaması aralık arama algoritması kullanılmaktadır. Öncelikle verilen x_1,x_2,y_1,y_2 aralık koordinatları tarafından kapsanan hücreler belirlenir. $(i_1,i_2=0,1,2,....n)$ $(j_1,j_2=0,1,2,...m)$ ve $x_1 \le \frac{Sn}{n} i_1$, $y_1 \le \frac{Sm}{m} j_1$, $x_2 \le \frac{Sn}{n} i_2$, $y_2 \le \frac{Sm}{m} j_2$ koşulunu sağlayan i_1,i_2,j_1,j_2 indis çiftleri

arasında kalan $h_1, h_2, ..., h_l$ hücreleri belirlenir. Her bir hücrenin i,j indis değerleri için $z=f(i,j)$ fonksiyonu kullanılarak Z değerleri belirlenmektedir. Belirlenen Z değerleri birincil anahtar olarak B+ ağacına iletilir ve bu değerlere karşılık gelen nesnelerin disk üzerindeki yerleri belirlenerek kullanıcıya nesnelerle ilgili bilgiler iletilmektedir. B+ ağacı tarafından anahtar değerlere karşılık herhangi bir nesneye ulaşılamıyorsa "kayıt yok" hata mesajı verilir.

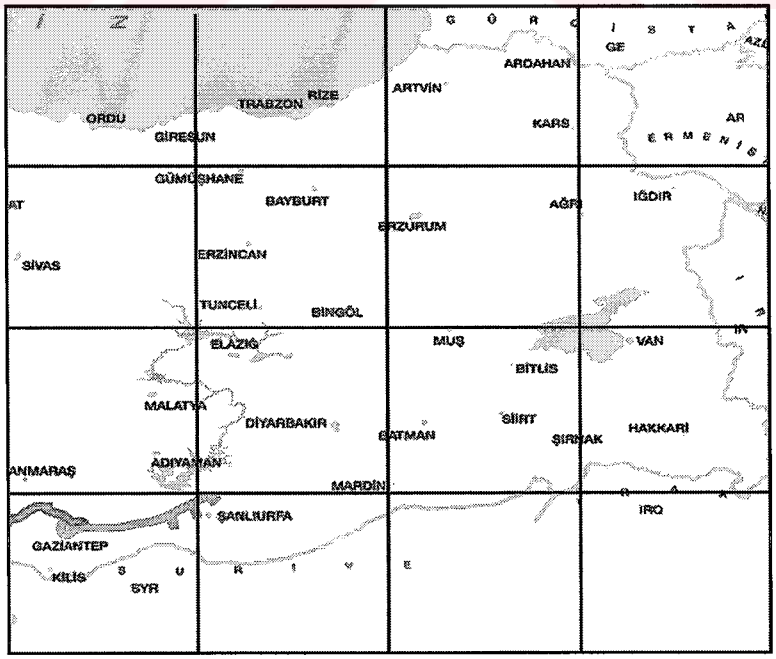


z	Nesne Adı	diğer
5	A	
12	B	

Şekil 3.29. Z sıralamasında aralık sorgulama

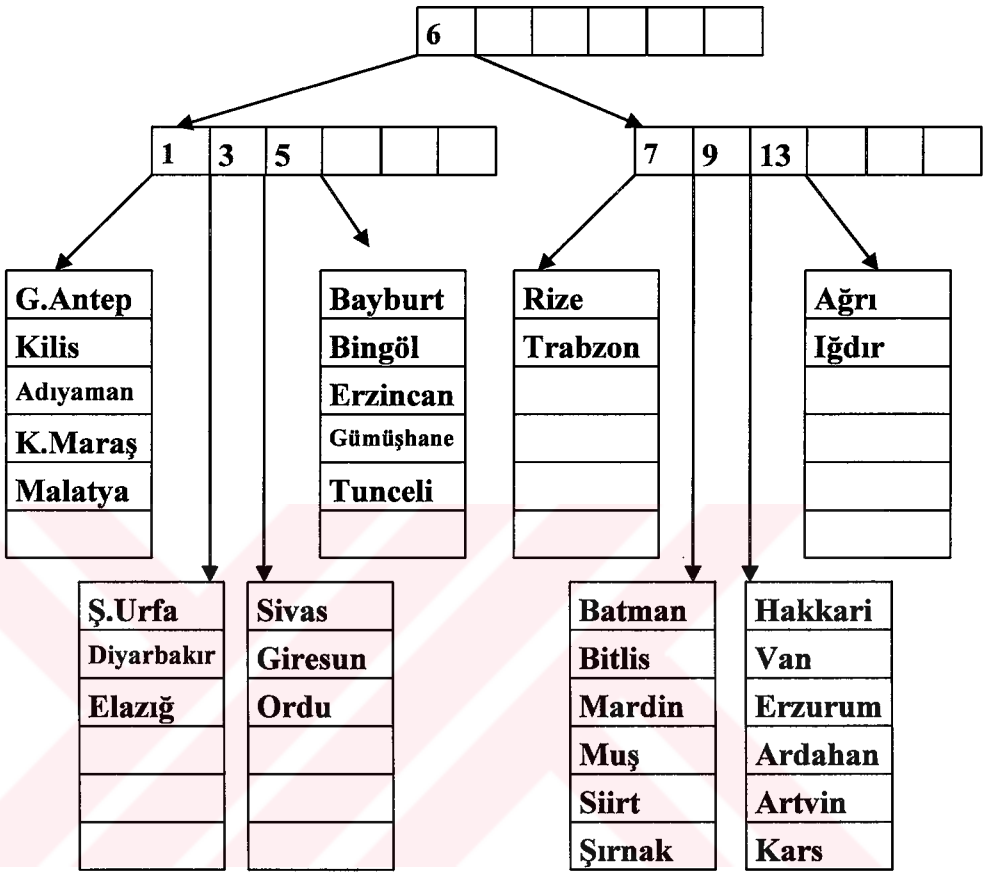
3.1.1.4.4. Örnek

Z sıralaması sorgulama algoritmalarını uygulayabilmek için çoklu ortam verisi olarak önemli yerleri ve yollarını içeren bir Türkiye haritası kullanılmaktadır.



Şekil 3.30. Örnek çoklu ortam verisi

Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil eder. Arama uzayının başlangıç noktası $(x_0=0,y_0=0)$ ve sınırları (160,160)'dır. Arama uzayı 4x4=16 adet eşit hücreye bölünmüştür. Hücrelerin kayıt kapasitesi altı olarak belirlenmiştir.

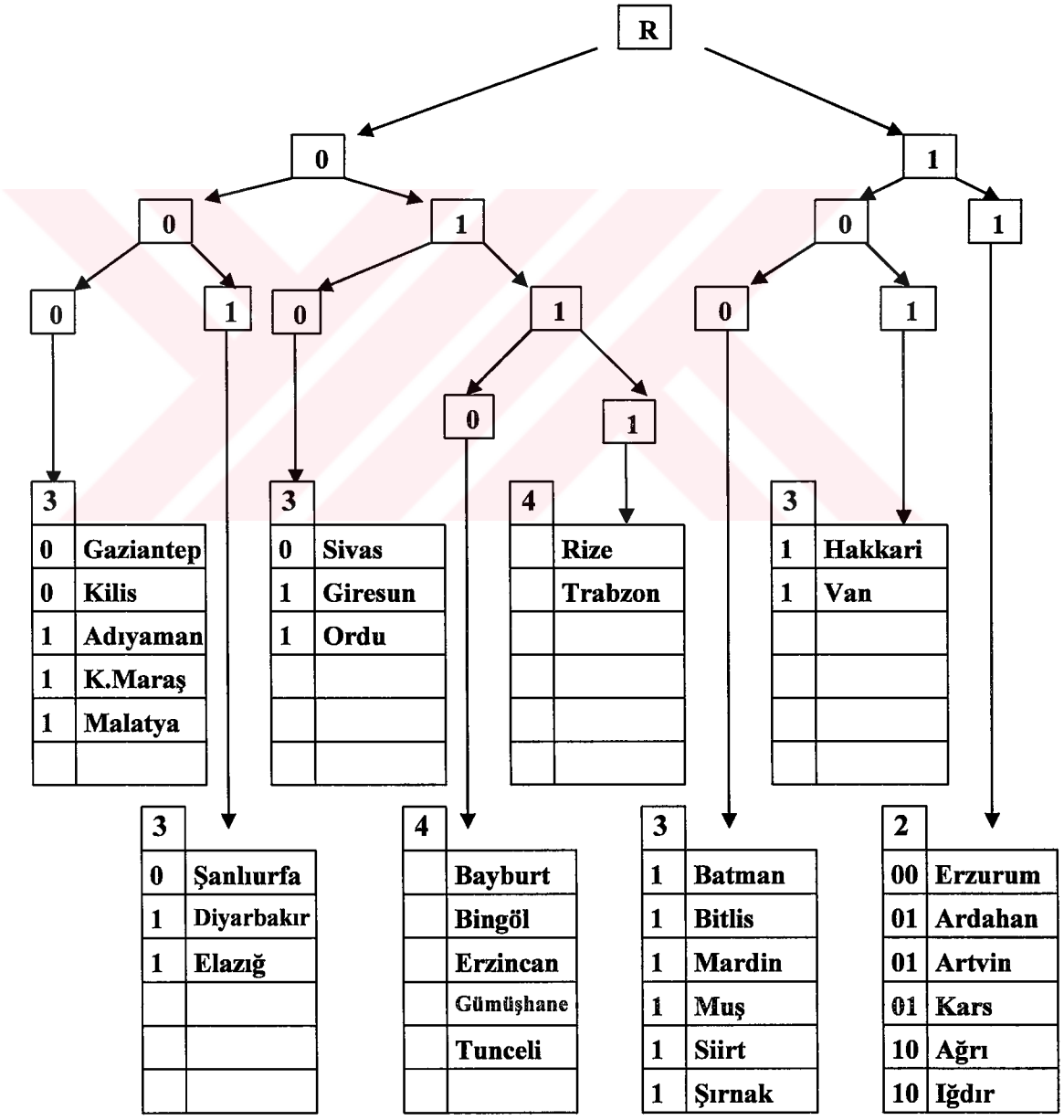


Şekil 3.31. Z sıralaması B+ ağaç yapısı

3.1.1.5. Z sıralaması yönteminde genişletilmiş hashing kullanımı

Z sıralaması belli bir ızgara çözünürlüğüne göre bölünmüş veri uzayındaki hücreleri numaralandırır. Bunun sonucunda her bir hücreye karşılık gelen numaraya Z değeri denilmektedir. Elde edilen bu Z değerlerinin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın Z sıralaması değerlerine göre veriler disk üzerine sırayla yerleştirilebilirler. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır.

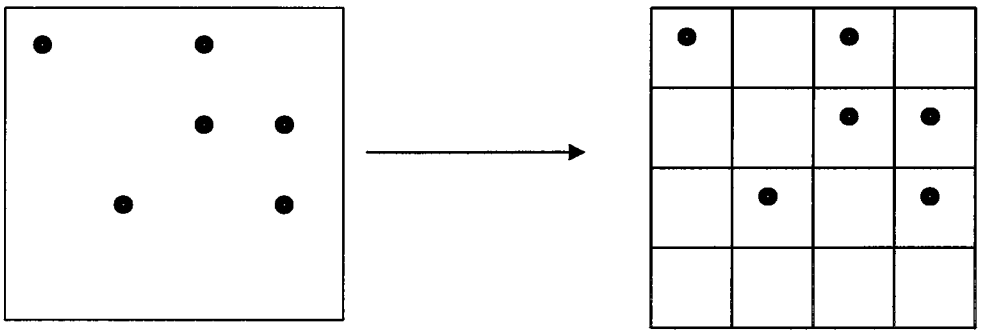
Z sıralaması yönteminde genişletilmiş hashing kullanımı erişim süresini azaltmaya ve ortak özellikteki verileri daha iyi gruplandırmaya yönelik bir yaklaşımdır. Tek boyuta indirgenen veri genişletilmiş hashing kullanılarak disk üzerine yerleştirilir. Genişletilmiş hashing yöntemi, benzer ve yakın anahtar değerlere sahip uzaysal verileri ardıl disk sayfalarında saklamaktadır. Bu, sorgulama algoritmalarının etkinliğini arttırmakta ve kayda erişim süresini kısaltmaktadır. Özellikle aralık sorgulamalarında benzer özelliklere sahip nesnelerin yakın disk bölgelerinde bulunmaları, erişim süresini çok azaltmaktadır.



Şekil 3.32. Z sıralaması genişletilmiş hashing ağaç gösterimi

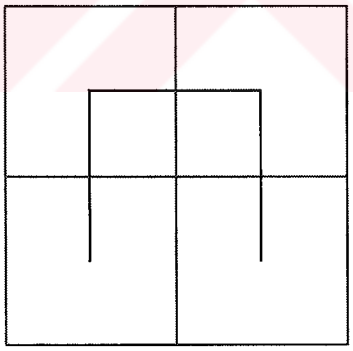
3.1.1.6. Hilbert Eğrisi

Hilbert eğrisi sıralaması çok boyutlu uzaysal verinin tek boyuta indirgenmesinde Z sıralaması metodundan farklı bir çözüm getirmektedir.



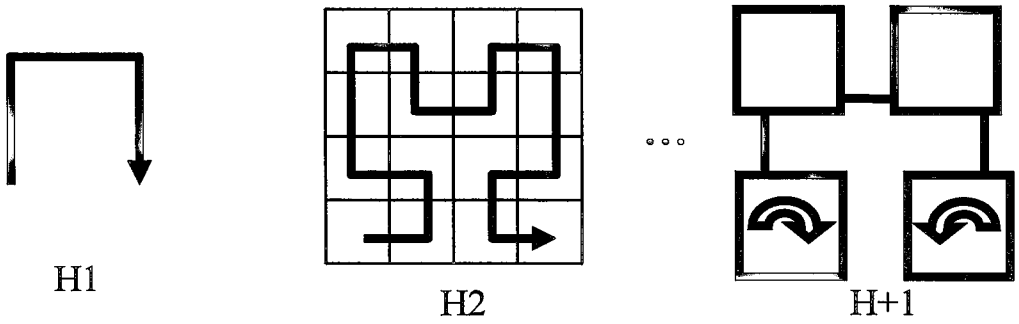
Şekil 3.33. Veri uzayının ızgaralara bölünmesi

n boyutlu verinin tek boyuta indirgenmesinde Hilbert eğrisi yöntemi veri uzayını belirli bir ızgara çözünürlüğüne göre statik olarak hücrelere bölmektedir. Daha sonra her bir hücreye bir numara vererek, numara sırasına göre disk üzerinde saklanmaktadır.



Şekil 3.34. 2x2 Boyutlu ızgara üzerinde Hilbert eğrisi

Hilbert eğrisi şekil 3.34'te 2x2 boyutlu ızgara üzerinde gösterilmiştir. Daha üst düzeylerde Hilbert eğrileri 0. ve 3. birim eğrilerin belirtilen yönlerde döndürülmesi köşelerin birleştirilmesiyle elde edilir. Birim eğri yinelenmiş bir biçimde daha üst düzeylere uygulanır. Şekilde Hilbert eğrisinin değişik düzeylerde uygulanışı gösterilmektedir. [Jagadish, 1997]



Şekil 3.35. 1,2 ve n. Düzey Hilbert eğrileri

Eğriyi takip eden hücreler artan sırayla numaralandırılırlar. Eğri sıralamasını *h=f(i,j)* şeklinde ifade etmek mümkündür. Herhangi bir hücrenin Hilbert eğri değerini hesaplamak için *i* ve *j* hücre indisleri kullanılmaktadır. Düzeylere bağlı tablolar kullanılarak *i,j* indislerine sahip hücrenin Hilbert eğri değeri hesaplanmaktadır.

Çizelge 3.1. Hilbert 3. düzey 1. tablo

x,y değeri	Hilbert değeri	Sonraki Tablo
0,0	0,0	2
0,1	0,1	1
1,0	1,1	1
1,1	1,0	3

Çizelge 3.2. Hilbert 3. Düzey 2. tablo

x,y değeri	Hilbert değeri	Sonraki Tablo
0,0	0,0	1
0,1	1,0	2
1,0	1,1	2
1,1	0,1	4

Çizelge 3.3. Hilbert 3. Düzey 3. tablo

x,y değeri	Hilbert değeri	Sonraki Tablo
0,0	1,1	4
0,1	0,1	3
1,0	0,0	3
1,1	1,0	1

Çizelge 3.4. Hilbert 3. Düzey 4. tablo

x,y değeri	Hilbert değeri	Sonraki Tablo
0,0	1,1	SON
0,1	1,0	SON
1,0	0,0	SON
1,1	0,1	SON

3 bit ile ifade edilen bir *S(6,4)* noktasının Hilbert eğri değeri hesaplanırken öncelikle *i* ve *j* değerlerinin ikili sayı sistemindeki karşılıkları kullanılmaktadır. *S* noktası *S(110,100)* ile ifade edilir. *S* noktasının koordinatlarını ifade eden *i* ve *j* değerlerinin en yüksek öncelikli *x* ve *y* bitleri ele alınır. 11 değeri 1.tabloda 10 değerine karşılık gelir. Daha sonraki bitler incelenirken 3 tablo kullanılacaktır. 10 değeri 3 tabloda 00 değerine karşılık gelir. Daha sonraki bitler incelenirken 3 tablo kullanılacaktır. 00 değeri 3 tabloda 11 değerine karşılık gelir. Elde edilen değerler birleştirildiğinde 100011=56 Hilbert eğri değeri elde edilir. Bit sayısı arttıkça kullanılan tablolar da değişmektedir.

Hücrelerin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın eğri değerlerine göre veriler disk üzerine sırayla yerleştirilebilirler. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır. Elde edilen değerler birincil anahtar olarak kullanılır ve verinin disk üzerine yerleştirilmesi B+ ağaç metoduyla gerçekleştirilmektedir.

Hilbert eğrisi değerinin disk üzerine yerleştirilmesinde kullanılan B+ ağaç yönteminin tercih edilmesindeki en büyük etmen, bu metodun çoğu yaygın sistemlerde hali hazırda bulunması, hata bakımından temiz olmasıdır. Hilbert eğrisi, Z sıralaması ile

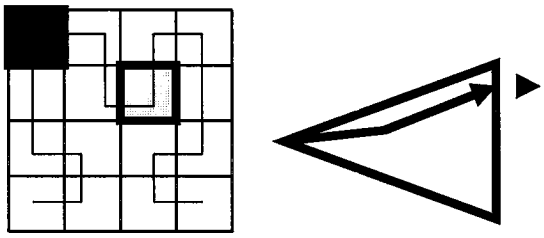
karşılaştırılacak olursa, verileri Z sıralamasından daha iyi bir biçimde kümelediği görülmektedir.

3.1.1.6.1. Hilbert eğrisi ekleme algoritması

n x m ızgara çözünürlüğüne *Sn, Sm* sınır değerlerine sahip bir veri uzayına *S(x,y)* koordinatlarına sahip bir nokta eklenmek istendiğinde hilbert eğrisi ekleme algoritması kullanılmaktadır. Öncelikle verilen *x,y* koordinatlarından noktanın dahil olacağı hücrenin indis değeri hesaplanır. $x \le \frac{Sn}{n} i$ ve $x \le \frac{Sm}{m} j$ koşulunu sağlayan *i,j* değerlerinden en küçüğü hücrenin indisi olarak belirlenir. *h=f(i,j)* fonksiyonu kullanılarak indis değeri verilen hücrenin Hilbert eğri değeri hesaplanır. Belirlenen değerler B+ ağaç yapısı tarafından birincil anahtar olarak kullanılır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlenmektedir.

3.1.1.6.2. Hilbert eğrisi nokta arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında, *S(x,y)* koordinatlarına sahip bir nokta aranırken hilbert eğrisi nokta arama algoritması kullanılmaktadır. Bu algoritma verilen *x,y* koordinatlarına göre nesnenin disk üzerindeki yerini belirler ve sonuç olarak nesneye ait özellikleri kullanıcıya döndürür. Öncelikle verilen *x,y* koordinatlarından noktanın içinde bulunabileceği olası hücrenin indis değeri hesaplanır. *(i=0,1,2,...n)*, *(j=0,1,2,...m)* ve $x \le \frac{Sn}{n} i$, $x \le \frac{Sm}{m} j$ koşulunu sağlayan en küçük *i,j* değerleri hücrenin indisi olarak belirlenir. *h=f(i,j)* fonksiyonu kullanılarak indisi verilen hücrenin Hilbert eğri değeri hesaplanır. Belirlenen değer B+ ağaç yapısı tarafından kullanılarak disk üzerinde nesne aranmaktadır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlendiğinde nesne ve özellikleri kullanıcıya bildirilmektedir. Aksi taktirde "kayıt yok" hata mesajı verilmektedir.

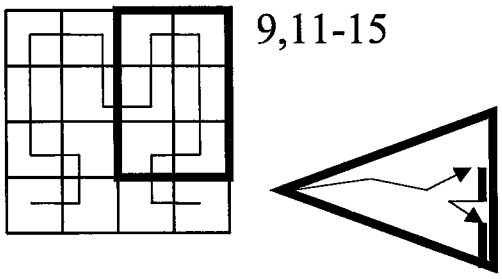


Hilbert	Nesne Adı	diğer
5	A	
12	B	

Şekil 3.36. Çoklu Ortam verilerinin diskte tutuluş biçimi

3.1.1.6.3. Hilbert eğrisi aralık arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında $P(x_1,y_1;x_2,y_2)$ penceresi tarafından kapsanan nesneler aranırken hilbert eğrisi aralık arama algoritması kullanılmaktadır. Öncelikle verilen x_1, x_2, y_1, y_2 aralık koordinatları tarafından kapsanan hücreler belirlenir. $(i_1,i_2=0,1,2,....n)$, $(j_1,j_2=0,1,2,...m)$ ve $x_1 \le \frac{Sn}{n} i_1$, $y_1 \le \frac{Sm}{m} j_1$, $x_2 \le \frac{Sn}{n} i_2$, $y_2 \le \frac{Sm}{m} j_2$ koşulunu sağlayan i_1, i_2, j_1, j_2 indis çiftleri arasında kalan h_1, h_2, ..., h_l hücreleri belirlenir. Her bir hücrenin *i,j* indis değerleri için $h=f(i,j)$ fonksiyonu kullanılarak Hilbert eğri değerleri belirlenir. Belirlenen değerler birincil anahtar olarak B+ ağacına iletilir ve bu değerlere karşılık gelen nesnelerin disk üzerindeki yerleri belirlenerek kullanıcıya nesnelerle ilgili bilgiler iletilmektedir. B+ ağacı tarafından anahtar değerlere karşılık herhangi bir nesneye ulaşılamıyorsa "kayıt yok" hata mesajı verilmektedir.

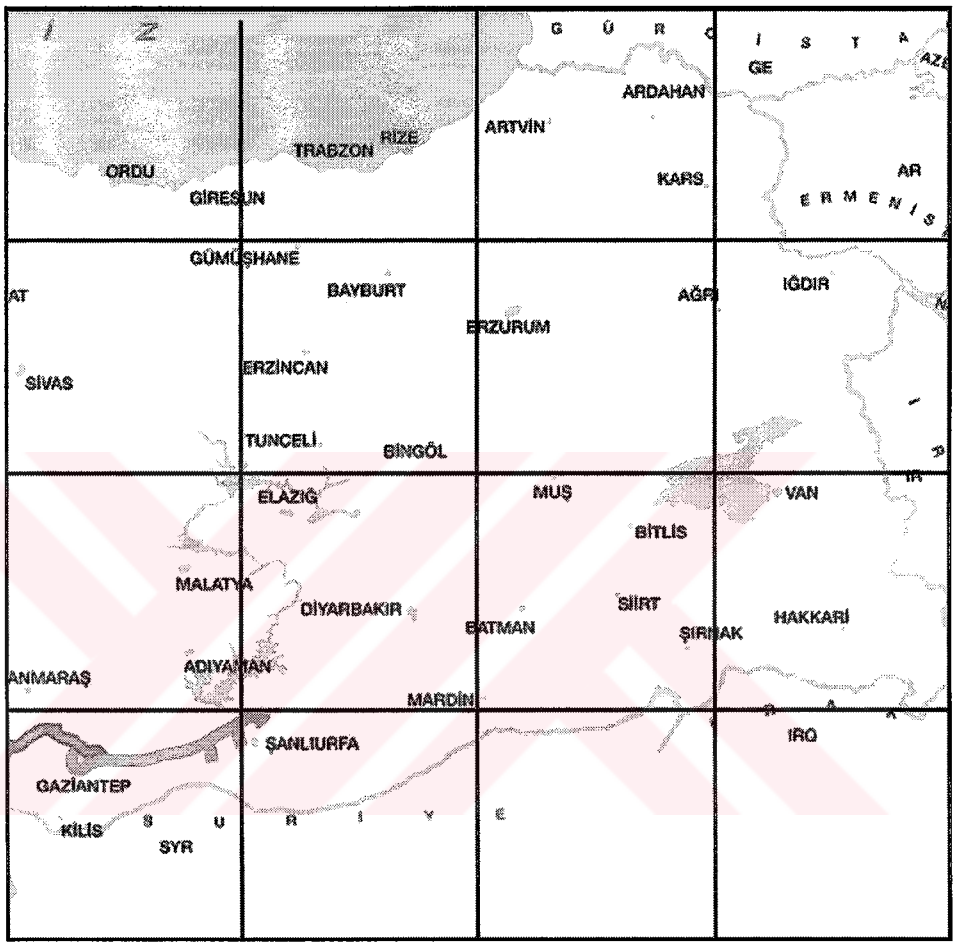


Hilbert	Nesne Adı	diğer
5	A	
12	B	

Şekil 3.37. Hilbert eğrisi aralık sorgulama

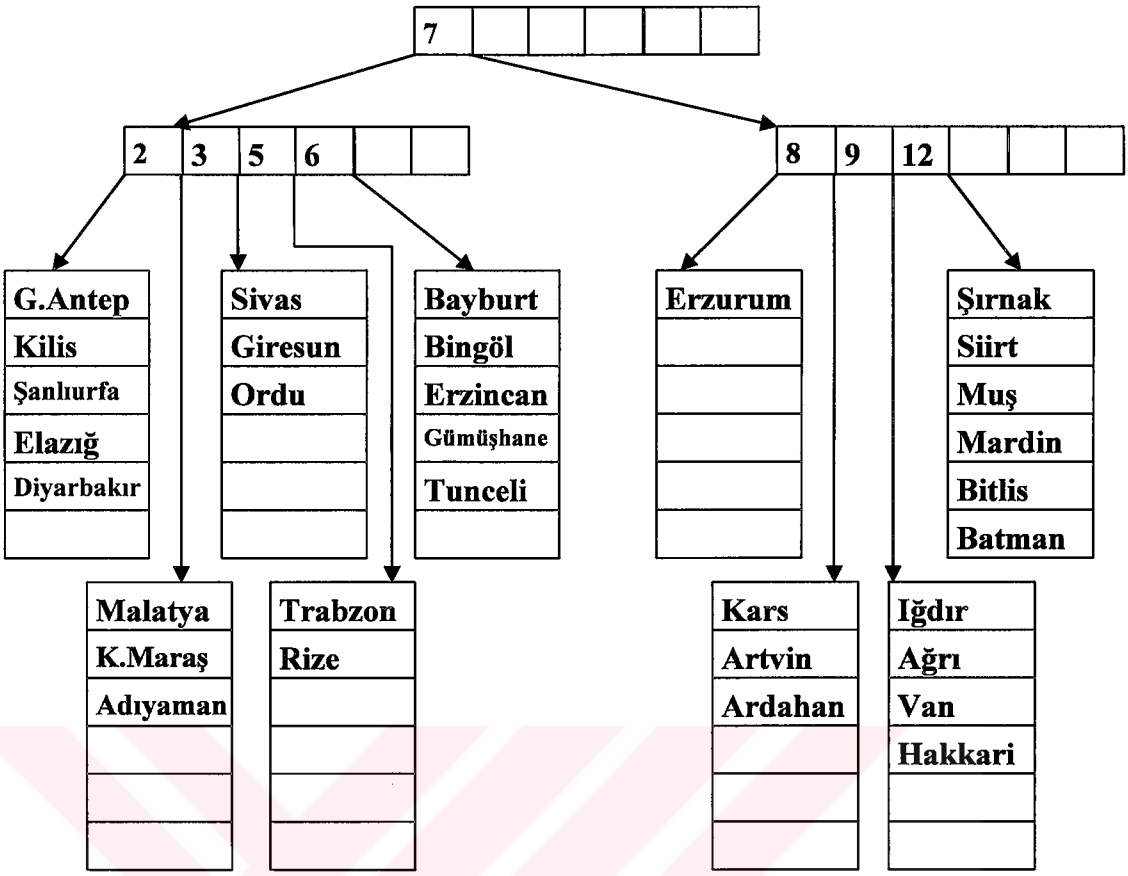
3.1.1.6.4. Örnek

Hilbert eğrisi sorgulama algoritmalarını uygulayabilmek için Çoklu ortam verisi olarak önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.



Şekil 3.38. Örnek Çoklu Ortam Verisi

Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil eder. Arama uzayının başlangıç noktası ($x_0=0,x_0=0$) ve sınırları (160,160)'dır. Arama uzayı 4x4=16 adet eşit hücreye bölünmüştür. Hücrelerin kayıt kapasitesi altı olarak belirlenmiştir.



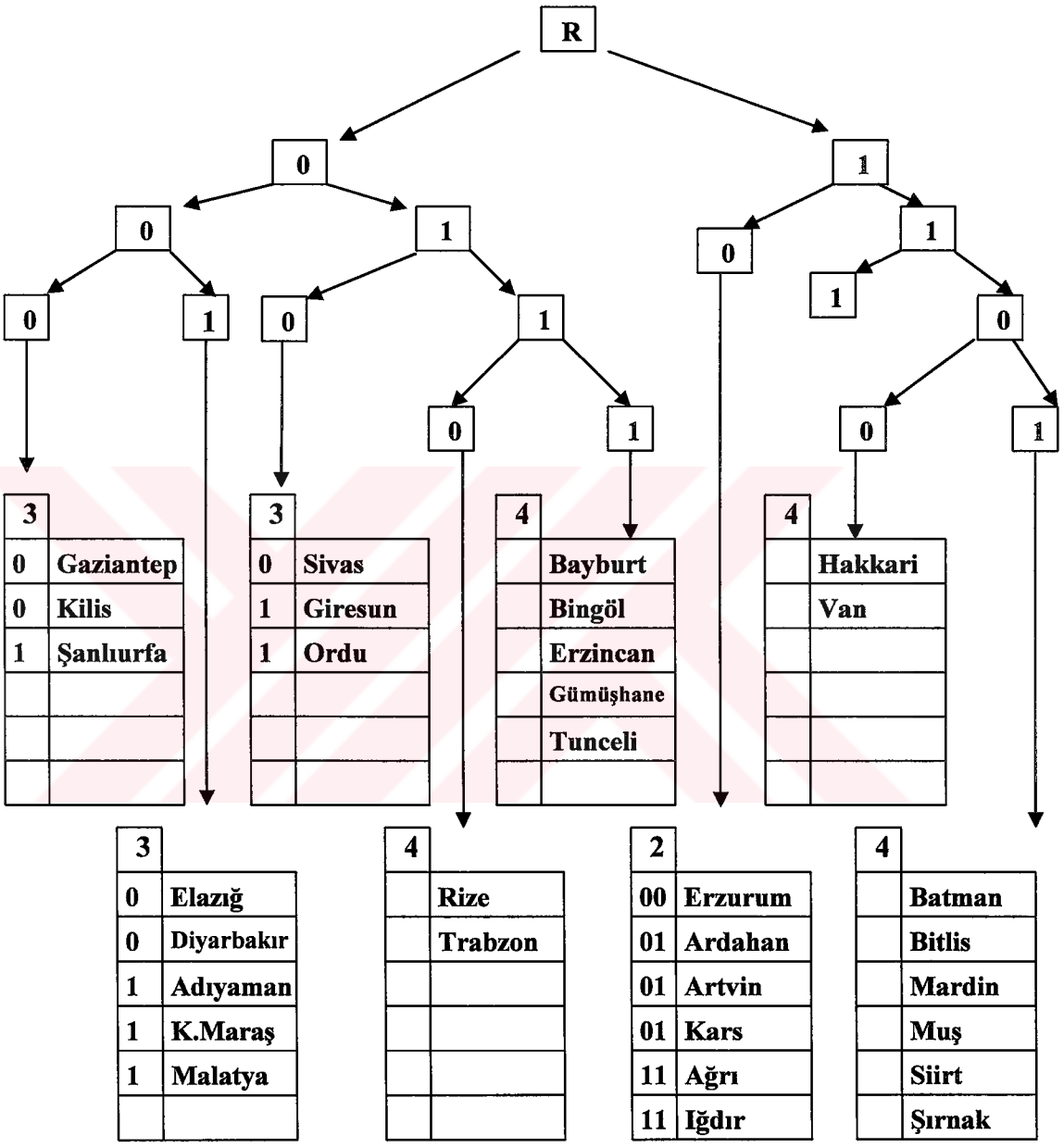
Şekil 3.39. Hilbert eğrisi B+ ağaç yapısı

3.1.1.7. Hilbert eğrisi yönteminde genişletilmiş hashing kullanımı

Hilbert eğrisi belli bir ızgara çözünürlüğüne göre bölünmüş veri uzayındaki hücreleri numaralandırmaktadır. Bunun sonucunda her bir hücreye karşılık gelen numaraya hilbert değeri denir. Elde edilen bu hilbert değerlerinin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın hilbert eğrisi değerlerine göre veriler disk üzerine sırayla yerleştirilebilirler. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır.

Hilbert eğrisi yönteminde genişletilmiş hashing kullanımı erişim süresini azaltmaya ve ortak özellikteki verileri daha iyi gruplandırmaya yönelik bir yaklaşımdır. Tek boyuta indirgenen veri genişletilmiş hashing kullanılarak disk üzerine yerleştirilmektedir. Genişletilmiş hashing yöntemi, benzer ve yakın anahtar değerlere sahip uzaysal verileri ardıl disk sayfalarında saklamaktadır. Bu, sorgulama

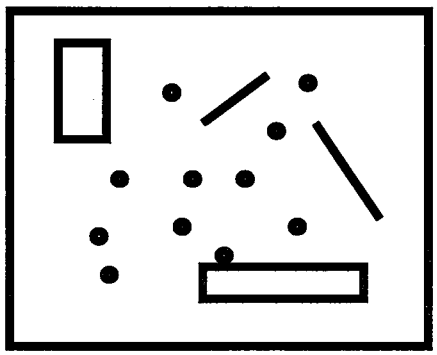
algoritmalarının etkinliğini arttırmakta ve kayda erişim süresini kısaltmaktadır. Özellikle aralık sorgulamalarında benzer özelliklere sahip nesnelerin yakın disk bölgelerinde bulunmaları, erişim süresini çok azaltmaktadır.



Şekil 3.40. Hilbert eğrisi genişletilmiş hashing ağaç gösterimi

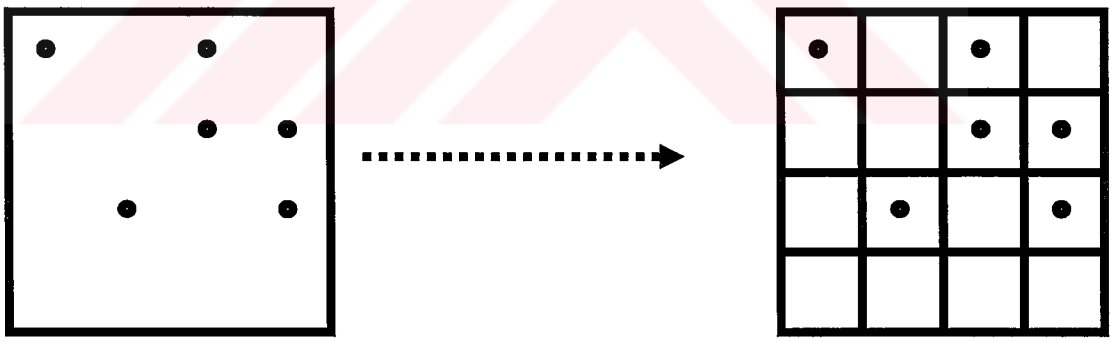
3.1.1.8. Gray Kodu Eğrisi

Gray kodu eğri sıralaması çok boyutlu uzaysal verinin tek boyuta indirgenmesine Z sıralaması ve Hilbert eğrisi metodundan farklı bir çözüm getirmektedir.



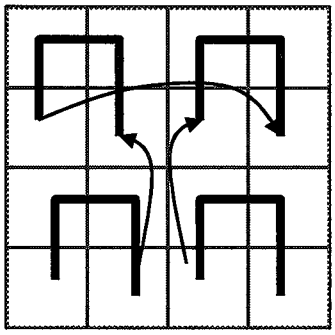
Şekil 3.41. İki boyutlu veri uzayı

n boyutlu verinin tek boyuta indirgenmesinde Gray kodu yöntemi veri uzayını belirli bir ızgara çözünürlüğüne göre statik olarak hücrelere bölmektedir. Daha sonra her bir hücreye bir numara vererek, numara sırasına göre disk üzerinde saklanmaktadır.



Şekil 3.42. Veri uzayının ızgaralara bölünmesi

Gray kodu şekil 3.42.'de 2x2 boyutlu ızgara üzerinde gösterilmektedir. Birim eğri yinelenen bir şekilde üst düzeylere uygulanmaktadır.



Şekil 3.43. Gray kodu eğrisi

Eğriyi takip eden hücreler artan sırayla numaralandırılırlar. Bu hücrelerin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın eğri değerlerine göre veriler disk üzerine sırayla yerleştirilebilir. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır. Elde edilen değerler birincil anahtar olarak kullanılır ve verinin disk üzerine yerleştirilmesi B+ ağaç metoduyla gerçekleştirilmektedir. [Faloutsos, 1986]

Gray kodu değerinin disk üzerine yerleştirilmesinde B+ ağaç yönteminin kullanılmasında en önemli etmen bu metodun çoğu yaygın sistemlerde hali hazırda bulunması, hata bakımından temiz olasıdır. Gray kodu benzer yöntemlerle karşılaştırıldığında verileri Z sıralamasından daha iyi bir biçimde kümelediği ,verilerin yoğunluğunun veri uzayının ortasında bulunması durumunda da hilbert eğrisinden daha etkin olduğu görülmektedir.

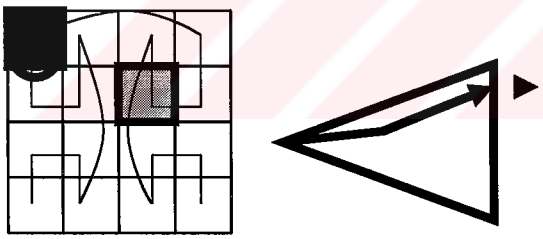
3.1.1.8.1. Gray kodu ekleme algoritması

n x m ızgara çözünürlüğüne *Sn, Sm* sınır değerlerine sahip bir veri uzayına *S(x,y)* koordinatlarına sahip bir nokta eklenmek istendiğinde Gray kodu ekleme algoritması kullanılmaktadır. Öncelikle verilen *x,y* koordinatlarından noktanın dahil olacağı hücrenin indis değeri hesaplanır. $x \le \frac{Sn}{n} i$ ve $x \le \frac{Sm}{m} j$ koşulunu sağlayan *i,j* değerlerden en küçüğü hücrenin indisi olarak belirlenir. *g=f(i,j)* fonksiyonu kullanılarak indis değeri verilen hücrenin Gray kodu eğri değeri hesaplanmaktadır. Belirlenen

değerler B+ ağaç yapısı tarafından birincil anahtar olarak kullanılmaktadır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlenmektedir.

3.1.1.8.2. Gray kodu nokta arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında, *S(x,y)* koordinatlarına sahip bir nokta aranırken Gray kodu nokta arama algoritması kullanılmaktadır. Bu algoritma verilen *x,y* koordinatlarına göre nesnenin disk üzerindeki yerini belirler ve sonuç olarak nesneye ait özellikleri kullanıcıya döndürür. Öncelikle verilen *x, y* koordinatlarından noktanın dahil bulunabileceği olası hücrenin indis değeri hesaplanır. *(i=0,1,2,...n), (j=0,1,2,...*m) ve $x \le \frac{Sn}{n} i$, $x \le \frac{Sm}{m} j$ koşulunu sağlayan en küçük *i,j* değerleri hücrenin indisi olarak belirlenir. *g=f(i,j)* fonksiyonu kullanılarak indisi verilen hücrenin Gray kodu değeri hesaplanmaktadır. Belirlenen değer B+ ağaç yapısı tarafından kullanılarak disk üzerinde nesne aranmaktadır. Nesnenin disk üzerindeki yeri B+ ağacı tarafından belirlendiğinde nesne ve özellikleri kullanıcıya bildirilir. Aksi taktirde "kayıt yok" hata mesajı verilmektedir.



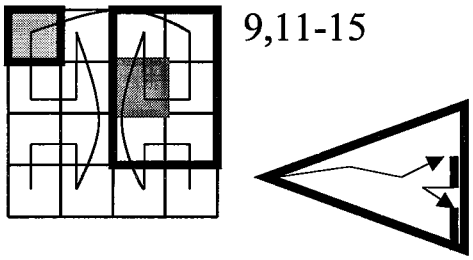
Gray Kodu	Nesne Adı	diğer
5	A	
12	B	

Şekil 3.44. Çoklu ortam verilerinin diskte tutuluş biçimi

3.1.1.8.3. Gray kodu aralık arama algoritması

n x m ızgara çözünürlüğüne ve *Sn, Sm* sınır değerlerine sahip bir veri uzayında $P(x_1,y_1;x_2,y_2)$ penceresi tarafından kapsanana nesneler aranırken gray kodu aralık arama algoritması kullanılmaktadır. Öncelikle verilen x_1, x_2, y_1, y_2 aralık koordinatları tarafından kapsanan hücreler belirlenir. $(i_1,i_2=0,1,2,....n)$, $(j_1,j_2=0,1,2,...m)$ ve $x_1 \le \frac{Sn}{n} i_1$, $y_1 \le \frac{Sm}{m} j_1$, $x_2 \le \frac{Sn}{n} i_2$, $y_2 \le \frac{Sm}{m} j_2$ koşulunu sağlayan i_1, i_2, j_1, j_2 indis

çiftleri arasında kalan h_1, h_2, ..., h_l hücreleri belirlenir. Her bir hücrenin i,j indis değerleri için $g=f(i,j)$ fonksiyonu kullanılarak Gray kodu eğri değerleri belirlenmektedir. Belirlenen değerler birincil anahtar olarak B+ ağacına iletilir ve bu değerlere karşılık gelen nesnelerin disk üzerindeki yerleri belirlenerek kullanıcıya nesnelerle ilgili bilgiler iletilir. B+ ağacı tarafından anahtar değerlere karşılık herhangi bir nesneye ulaşılamıyorsa "kayıt yok" hata mesajı verilmektedir.

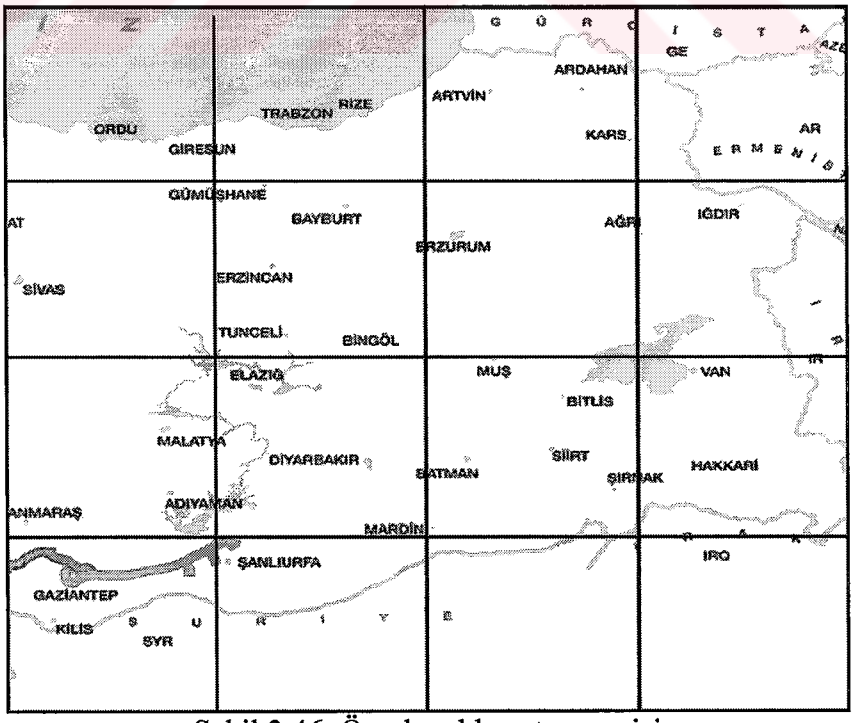


Gray Kodu	Nesne Adı	diğer
5	A	
12	B	

Şekil 3.45. Gray kodu aralık sorgulama

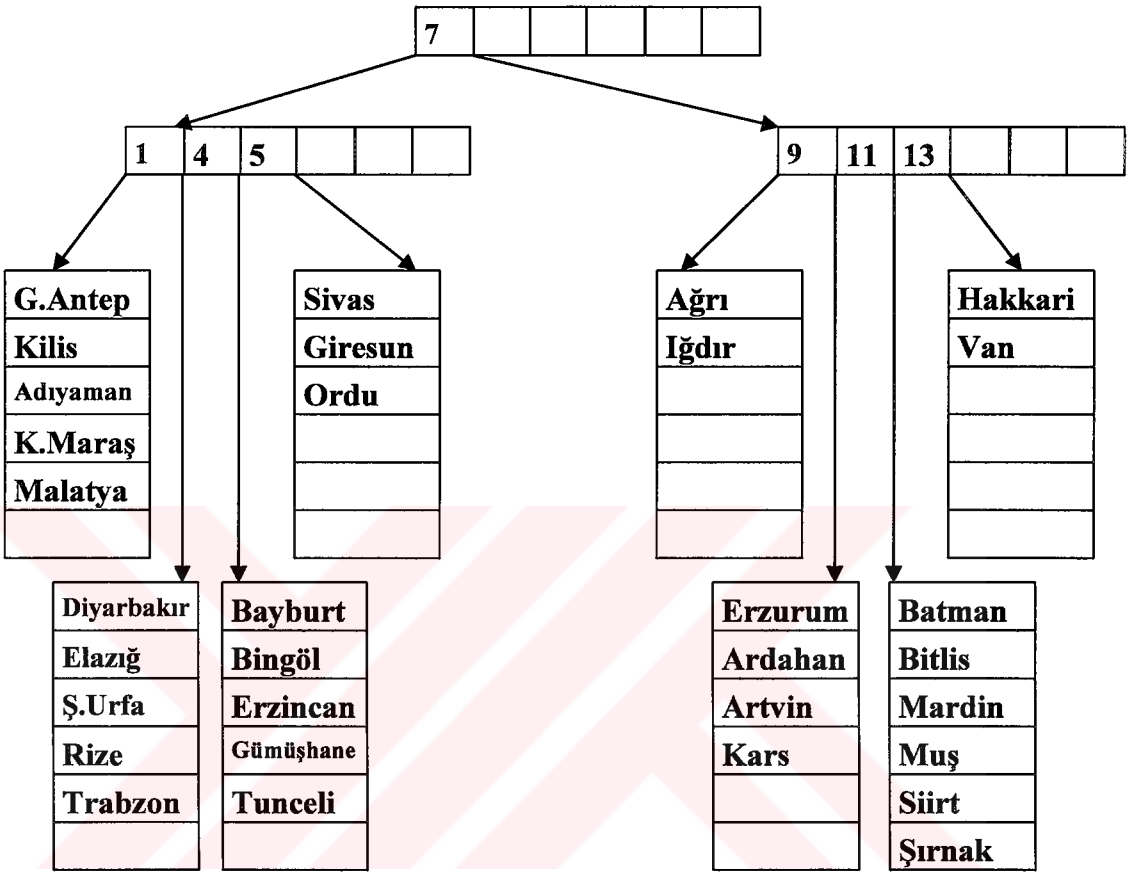
3.1.1.8.4. Örnek

Gray kodu sorgulama algoritmalarını uygulayabilmek için Çoklu ortam verisi olarak önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.



Şekil 3.46. Örnek çoklu ortam verisi

Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil eder. Arama uzayının başlangıç noktası $(X_0=0, Y_0=0)$ ve sınırları (160,160)'dır. Arama uzayı 4x4=16 adet eşit hücreye bölünmüştür. Hücrelerin kayıt kapasitesi 6 olarak belirlenmiştir.

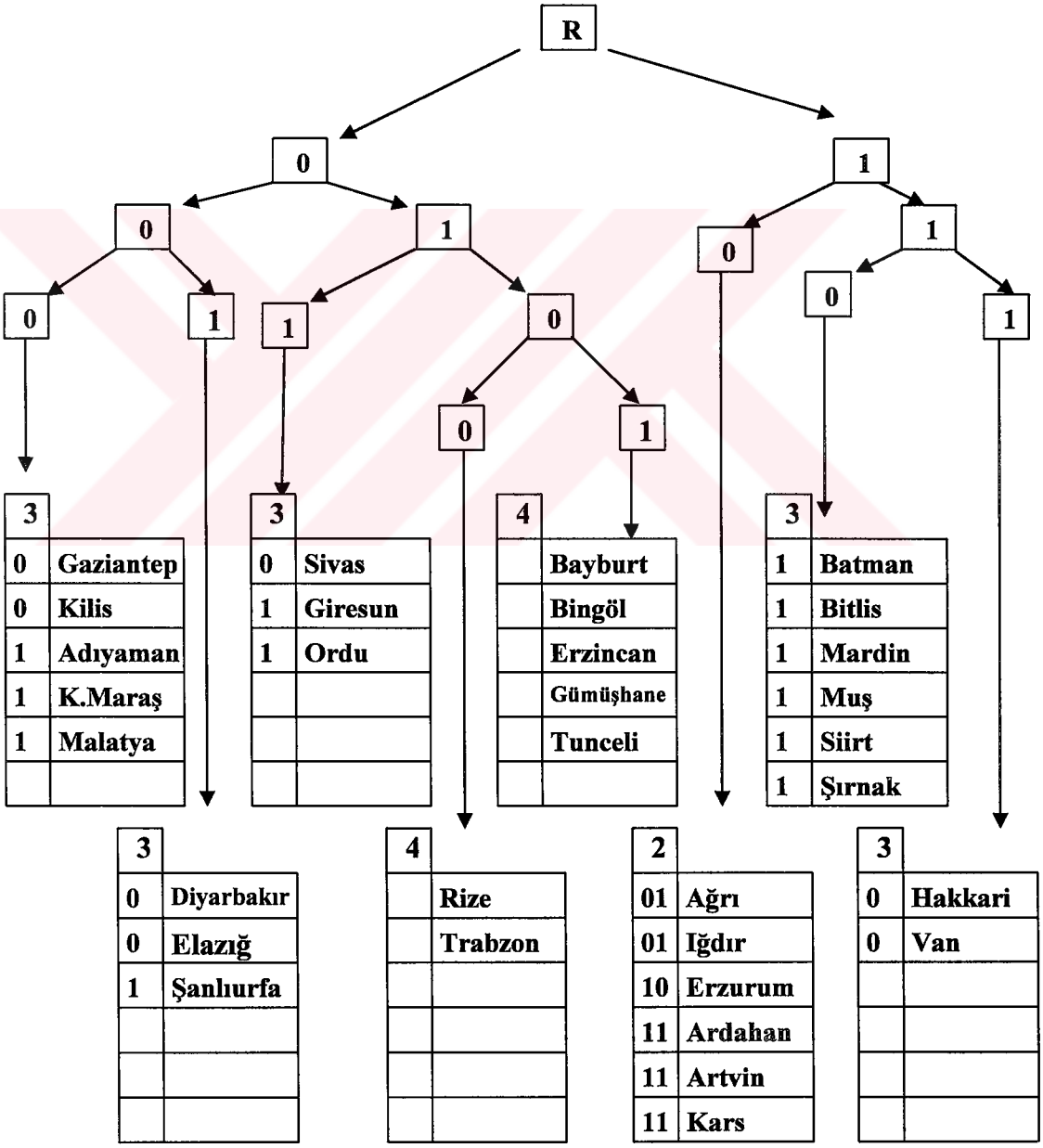


Şekil 3.47. Gray kodu eğrisi B+ ağaç yapısı

3.1.1.9. Gray kodu eğrisi yönteminde genişletilmiş hashing kullanımı

Gray kodu eğrisi belli bir ızgara çözünürlüğüne göre bölünmüş veri uzayındaki hücreleri numaralandırmaktadır. Bunun sonucunda her bir hücreye karşılık gelen numaraya Gray kodu değeri denilir. Elde edilen bu Gray kodu değerlerinin disk üzerinde saklanmasında da farklı yaklaşımlar bulunmaktadır. Herhangi bir erişim metodu kullanmaksızın Gray kodu eğrisi değerlerine göre veriler disk üzerine sırayla yerleştirilebilirler. Fakat her katmanda değişik iyileştirmeler içeren yöntem, etkinliğin daha da arttırılması amacıyla sıralı erişim metodu kullanmamaktadır.

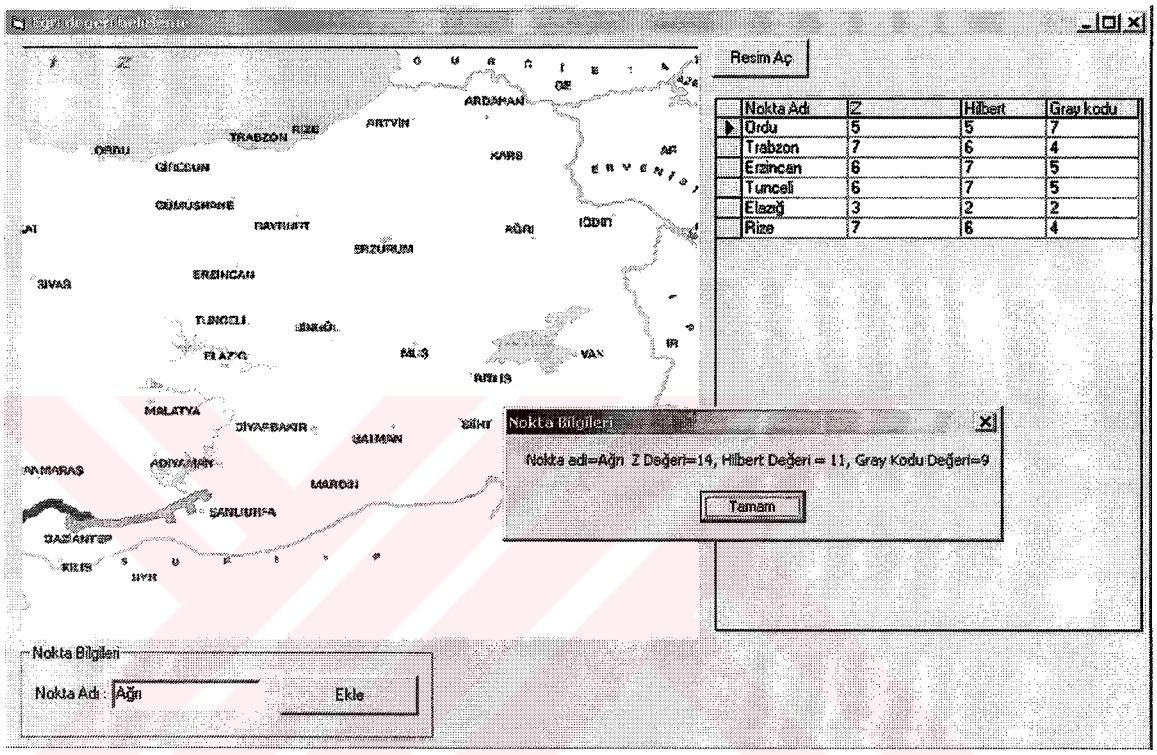
Gray kodu eğrisi yönteminde genişletilmiş hashing kullanımı erişim süresini azaltmaya ve ortak özellikteki verileri daha iyi gruplandırmaya yönelik bir yaklaşımdır. Tek boyuta indirgenen veri genişletilmiş hashing kullanılarak disk üzerine yerleştirilmektedir. Genişletilmiş hashing yöntemi, benzer ve yakın anahtar değerlere sahip uzaysal verileri ardıl disk sayfalarında saklamaktadır. Bu, sorgulama algoritmalarının etkinliğini arttırmakta ve kayda erişim süresini kısaltmaktadır. Özellikle aralık sorgulamalarında benzer özelliklere sahip nesnelerin yakın disk bölgelerinde bulunmaları, erişim süresini çok azaltmaktadır.



Şekil 3.48. Gray kodu eğrisi genişletilmiş hashing ağaç gösterimi

3.1.1.10. Eğri yöntemlerinin bilgisayar destekli uygulaması

Geliştirilen program Delphi 6.0 kullanılarak hazırlanmıştır. Bu program verilen .bmp uzantılı ve 320x320 çözünürlükteki resim üzerinde belirlenen bir noktanın Z sıralaması, Hilbert ve Gray kodu değerlerini bulmaktadır.



Şekil 3.49. Uygulama yazılımı ekran görüntüsü

Programda "Resim Aç" düğmesine basılarak resim dosyaları yüklenmektedir. Yüklenen Resim dosyası kullanıcıya gösterilmekte ve kullanıcıdan resim üzerinde bir nokta seçmesi ve bir nokta adı belirlemesi istenmektedir."Ekle" düğmesine basıldığında kullanıcıya nokta hakkında Z sıralaması, Hilbert ve Gray kodu değerleri gibi bilgilerin bulunduğu özet bir mesaj çıkmakta ve bu bilgiler program tarafından saklanmaktadır.

3.1.1.10.1 Program kodlarının açıklaması:

Aşağıdaki fonksiyon z-order değerlerini üretmektedir.

procedure tform1.z_order_str_olustur;

```
var i,j:integer;
begin
 z_order_str:='';
 for i:=0 to max-1 do
  for j:=0 to max-1 do
   if (nokta[i,j]=1) then
    z_order_str:=z_order_str+inttostr(birlestir(binary(j),binary(max-1-i)))+' ';
end;
```

Aşağıdaki fonksiyon gray code değerlerini üretmektedir.

```
procedure tform1.gray_olustur2;
var a,i,j:integer;
   tmp  :string;
begin
 gray_str2:='';
 for i:=0 to max-1 do
  for j:=0 to max-1 do
   if (nokta[i,j]=1) then
   begin
      tmp:=inttostr(birlestir(binary(j),binary(max-1-i)));
      tmp:=binary(strtoint(tmp));
      for a:=length(tmp) downto 1 do
      begin
        if tmp[a-1]='1' then
          if tmp[a]='0' then tmp[a]:='1' else tmp[a]:='0';
      end;
      tmp:=inttostr(decimal(tmp));
      gray_str2:=gray_str2+tmp+' ';
   end;
end;
```

Yukarıdaki iki fonksiyon değerleri koordinatları kullanarak bulmaktadır. Hilbert eğri değerlerini bulmak için ise daha önceden hazırlanmış olan hilbert kod tablosundaki değerlerle karşılaştırma yapılır. Bu tablo aşağıdaki fonksiyon tarafından hazırlanmaktadır.

```
procedure tform1.hilbert_olustur;
var
 x,y  :integer;
 n    :integer;
 boy  :integer;
 son  :integer;
begin
 n:=0;
 repeat
  boy:=variant(power(2,n));
  son:=variant(power(4,n+1));
  if (n mod 2)=0 then
  begin
   for x:=max-1 downto max-boy do
    for y:=boy to boy*2-1 do
      hilbert[x,y]:=variant(son*0.5)-1-hilbert[max-1-(y-boy),x-(max-boy)];
   for y:=0 to boy*2-1 do
    for x:=max-1-boy downto max-boy*2 do
      hilbert[x,y]:=son-1-hilbert[max-boy+(max-1-boy-x),y];
  end
  else
  begin
   for x:=max-1-boy downto max-boy*2 do
    for y:=0 to boy-1 do
      hilbert[x,y]:=variant(son*0.5)-1-hilbert[max-boy+y,(max-1-boy-x)];
   for x:=max-1 downto max-boy*2 do
    for y:=boy to boy*2-1 do
```

```
        hilbert[x,y]:=son-1-hilbert[x,boy-1-(y-boy)];
    end;
    n:=n+1;
  until(boy*2>=max)
end;
```

Program çalışmaya başladığı anda yukarıdaki fonksiyon çalıştırılarak hilbert kod tablosu oluşturulmaktadır.

3.1.2. Veri Kontrollü Yapılar

Uzay kontrollü yapılar çoklu ortam verilerini kontrol etmek için veri uzayını klasik erişim yöntemleriyle indekslenebilecek bir hale dönüştürürler. Veri kontrollü yapılar klasik erişim yöntemlerine dönüşüm yapmazlar. Veri kontrollü yapılar uzaysal özellikleri kullanarak verileri sınıflandırır. Veri kontrollü yapılardan R Tree nesneleri çok boyutlu bir ağaç yapısı kullanarak indeksler.

3.1.2.1. R Tree

R Tree, veri kontrollü yapılardan B+ ağacına benzer yükseklik dengeli bir ağaç yapısıdır. B+ ağaç yapısının *n* boyuta genişletilerek oluşmuş, çok boyutlu uzayda bir hacmi olan nesneleri dinamik olarak kontrol edebilen bir indeks yapısıdır. Diğer yapılardan en üstün yanı bir dönüşüm tekniği değil gerçek bir uzaysal erişim metodu olmasıdır. Uzaysal nesneleri dönüşümler yoluyla tek boyuta indirgeyip geleneksel erişim metotlarıyla indekslemek yerine nesneleri çok boyutlu bir ağaç yapısı kullanarak indekslemektedir. [Guttman, 1984]

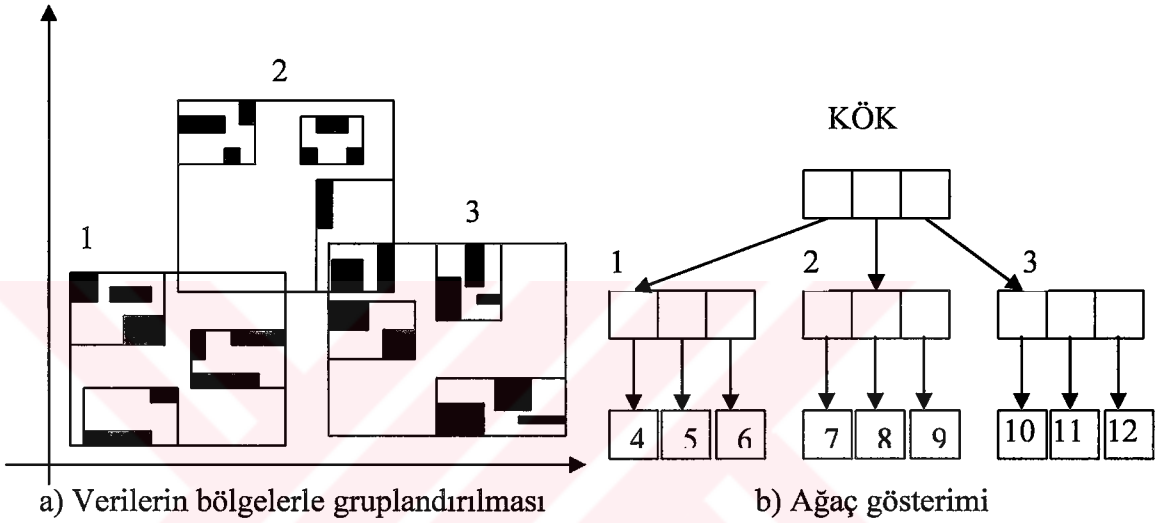
Antonin Guttman tarafından 1984 yılında sunulan yöntem çok boyutlu indekslemede bir çok yönteme (SR, SS, STR, TV, R+, R*) temel teşkil etmiştir. R Tree yönteminin dinamik bir indeks yapısı vardır, periyodik yeniden oluşumlar gerektirmez.

R Tree yapısında iki tip düğüm bulunmaktadır. Yaprak düğümler ve yaprak olmayan düğümler. Yaprak düğümler *(I,Nesne_P)* şeklinde nesneleri sınırlayan dikdörtgeni ifade eden $I(I_0,I_1,...I_{n-1})$ vektör bilgisine ve *Nesne_P* nesne işaretçisine sahiptirler. Yaprak olmayan düğümler ise *(I,cocuk_P)* şeklinde alt ağaçlarda bulunan

sınırlayıcı dikdörtgenleri sınırlayan dikdörtgenleri ifade eden $I(I_0,I_1,..I_{n-1})$ vektörel bilgisine ve *cocuk_P* alt çocuk düğümü işaretçisine sahiptirler.

R Tree'nin derecesi *(m,M)* ikilisi ile ifade edilir. R Tree yapısında her iki düğüm tipi en az $m<m/2$ ve en fazla M adet kayıt sayısına sahip olmalıdır.

R Tree ikincil bellek kullanımı için çok uygundur. R Tree'nin her bir düğümü ayrı disk sayfasında saklanmaktadır. Bu indeks yapısı belleğe sığmayacak kadar büyük olan veri tabanlarında R Tree yönteminin kullanılabilmesini uygun kılmaktadır.



a) Verilerin bölgelerle gruplandırılması b) Ağaç gösterimi

Şekil 3.50. R Tree

3.1.2.1.1. R Tree arama algoritması

R Tree arama algoritması verilen kök T düğümünden başlayarak arama dikdörtgeni S ile kesişen tüm indeks kayıtlarını bulmaya dayanmaktadır. İki adımdan oluşur.

1- Alt Ağaç Arama: Eğer T bir yaprak düğüm değil ise bu düğümde bulunan tüm E kayıtlarının S ile kesişen E_I sınırlayıcı dikdörtgenleri belirlenir. Tüm kesişen kayıtlar için *cocuk_P* işaretçisi tarafından belirtilen alt ağaçlar yinelene biçimde arama algoritmasına yönlendirilir.

2- Yaprak Düğüm Arama: T Eğer yaprak düğüm ise bu düğümde bulunan tüm E kayıtları kontrol edilir. E_I ile S arama dikdörtgeni kesişiyorsa E kaydını döndürür. Eğer kesişen bir kayıt yoksa "kayıt yok" mesajı döndürülür.

3.1.2.1.2. R Tree ekleme algoritması

R Tree Ekleme algoritması verilen *E* kaydının ağaç yapısına dahil edilmesine dayanmakta ve dört adımda gerçekleşmektedir.

1- Konum bulma: "Yaprak Seçme" algoritması çağırılarak *E* kaydının yerleştirileceği *L* yaprak düğümü belirlenir.

2- Yaprak Düğüme Kayıt Ekleme: Eğer *L* yaprağının *E* kaydını alabilecek yeterli kapasitesi varsa *E* kaydı yaprak düğüme eklenir. Diğer durumda "Düğüm Bölünme" algoritması çağırılır. *L* ve *LL* yaprak düğümleri elde edilir.

3- Değişiklikleri Üst Düzeylere Uygulamak: Bölünme sonunda oluşan *L* ve *LL* düğümlerinde "Ağaç Düzenleme" algoritması çağrılır.

4- Ağacın Büyüyerek Uzaması: Eğer düzenleme sonucunda kök düğüm bölünür ise çocukları bölünme sonucunda oluşan düğümler olan yeni bir kök düğüm oluşturulur.

3.1.2.1.3. R Tree yaprak seçme algoritması

Bu algoritma verilen bir *E* kaydının uygun olduğu yaprağı belirlemektedir ve dört adımda gerçekleşmektedir.

1- Başlangıç: N düğümünü kök düğüm olarak alınır.

2- Yaprak Kontrol : Eğer *N* Yaprak düğüm ise *N* düğümünü sonuç olarak döndürülür.

3- Alt Ağaç Seçme: N düğümünde *E* kaydını içine aldığı taktirde *F_I* sınırlayıcı dikdörtgenlerinden en az alan büyümesini gerektiren *F* alt düğümü seçilir.

4- Bir Yaprak Düğüme Ulaşıncaya Dek Azaltma: F_P tarafından gösterilen düğüm işaretçi değerini *N*'ye atanır ve 2.Adıma geçilir.

3.1.2.1.4. R Tree ağaç düzenleme algoritması

Yaprak düğüm *L*'den kök düğüme kadar sınırlayıcı dikdörtgen *I* değerlerini ve düğüm bölünmelerini düzenlemektedir. Beş adımda gerçekleşir.

1- Başlangıç: N=L yap, Eğer daha önce *L* bölünmüş ise *NN* değerine ikinci düğümün adresini atanır.

2- Bitiş Kontrolü: Eğer *N* kök düğüm ise algoritma bitirilir.

3- Üst düğümde Sınırlayıcı Dikdörtgen Düzenleme: P düğümü *N* düğümünün üst düğümü olduğu taktirde *P* düğümünde bulunan E_n kaydı bu düğümü ifade eder. *N* düğümünde bulunan tüm kayıtları kapsayan sınırlayıcı dikdörtgen değeri E_n_I değeri güncellenir.

4- Düğüm bölünmelerini ayarla: NN kaydını gösteren E_{NN} ve E_n_I değerleri *P* düğümünde oluşturulur. *P* düğümünün kapasitesi dolu ise Bölünme algoritması çalıştırılır ve *P , PP* alt düğümleri oluşturulur.

5- Bir sonraki seviyeye davam: N=P ve *NN=PP* atamaları yapılır, ikinci adımdan devam edilir.

3.1.2.1.5. R Tree silme algoritması

E kaydının indeks değerinin R tree'den çıkartılması işlemidir. Dört adımda gerçekleşmektedir.

1- Düğüm Bul: "Yaprak Bulma" algoritması çağırılarak *E* kaydını içinde bulunduran düğüm belirlenir. Eğer bulunmaz ise algoritma sonlandırılır.

2- Kaydı Silme: E kaydını *L* düğümünden silinir.

3- Değişikleri Uygula: L düğümü için Ağaç Yoğunlaştırma algoritması çalıştırılır.

4- Ağacı Kısaltma: Eğer düzenleme sonucunda kök düğümün sadece tek çocuğu kalmışsa bu düğümü kök düğüm haline getirilir.

3.1.2.1.6. R Tree yaprak bulma algoritması

Verilen kök düğüm *T* değerinden başlayarak R tree indeks yapısında E kaynının bulunduğu yaprağı belirlemektedir. İki adımda gerçekleşir.

1- Alt ağaçları arama: Eğer *T* yaprak düğüm değil ise E_I ile kesişen tüm F_I sınırlayıcı dikdörtgenlere ait *F* kaydını gösteren her bir F_P değerleri için *E* kaydı bulunana kadar yaprak bulma algoritması uygulanır.

2- Kayıt için yaprak düğümün aranmasa: Eğer *T* yaprak düğüm ise her bir kayıt kontrol edilir. Eğer *E* kaydı bulunursa *T* döndürülür.

3.1.2.1.7. R Tree ağaç yoğunlaştırma algoritması

L kaydının bir düğümden silinmesi durumunda sınırlayıcı dikdörtgenlerin güncelleştirilmesi ve eğer yeterli kayıt kapasitesine sahip olmayan bir düğüm içindeki kayıtların yeniden yerleştirilmesi işlemlerini gerçekleştirmektedir. Altı adımda gerçekleşir.

1- Başlangıç Durumu: N=L yap, *Q* yok edilecek olan düğümler kümesini gösterir.

2- Baba düğüm Bulma: Eğer *N* kök düğüm ise 6.adıma geç. Aksi taktirde *N* düğümünün babası olarak *P* düğümünü belirlenir.

3- Düşük kapasiteli düğümlerin silinmesi: Eğer *N* düğümünde *m* adetten daha az sayıda kayıt bulunuyorsa E_N kaydı *P* düğümünden silinir ve *N* düğümü *Q* kümesine dahil edilir.

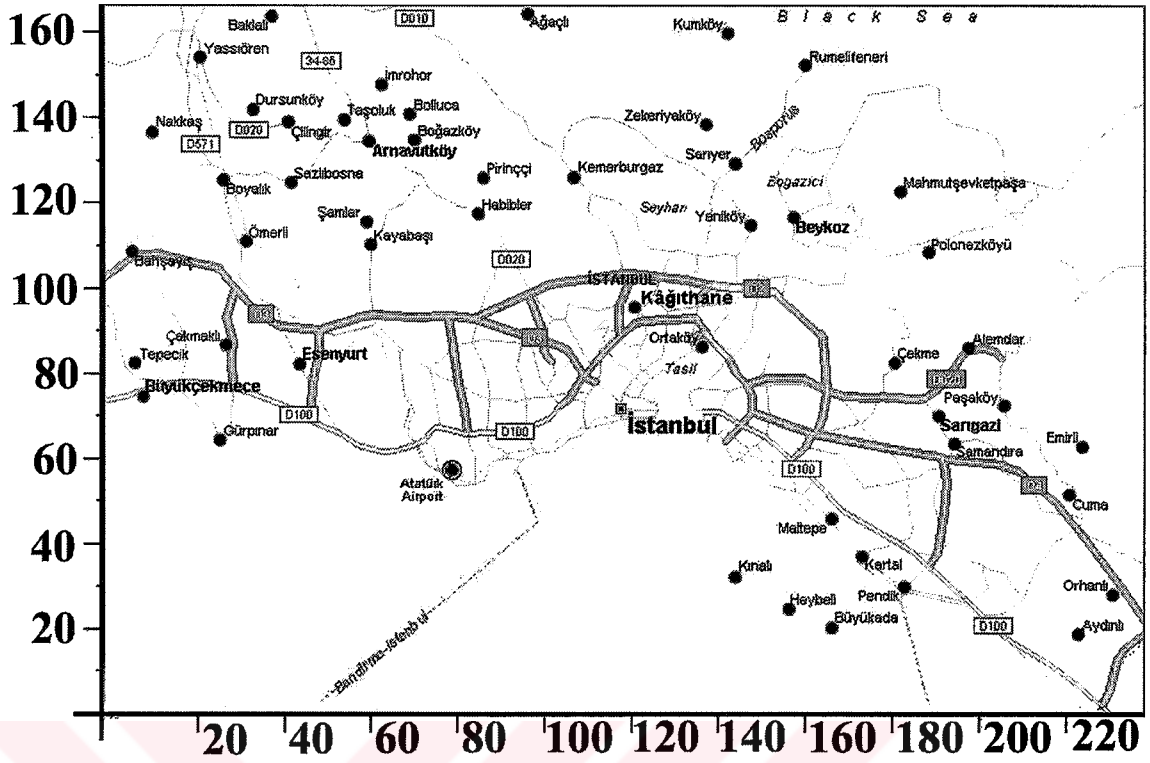
4- Sınırlayıcı dikdörtgenlerin ayarlanması: Eğer *N* yok edilmemiş ise *N* düğümündeki tüm kayıtları kapsayacak biçimde E_N_I değeri düzenlenir.

5- Bir üst seviyeye geçme: N=P yapılır ve 2. adıma geçilir.

6- Boşta Kalan Kayıtların Tekrardan Eklenmesi: Q kümesinde bulunan tüm kayıtlar ekleme algoritması kullanılarak ağaç yapısına tekrardan dahil edilir.

3.1.2.1.8. Örnek

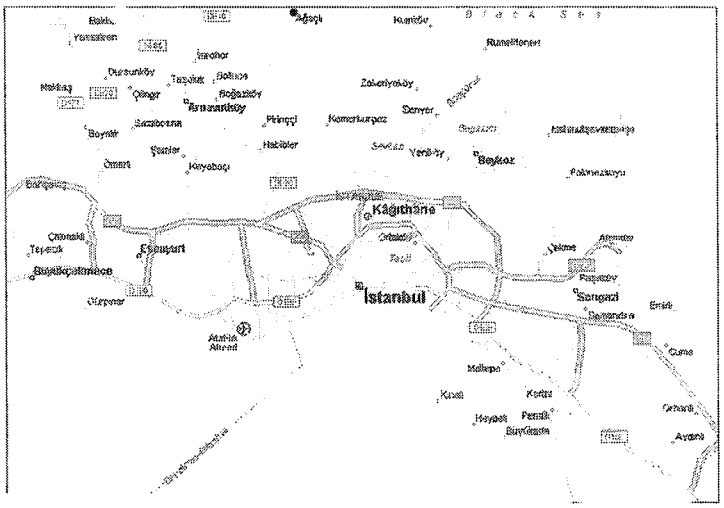
R Tree sorgulama algoritmalarını uygulayabilmek için çoklu ortam verisi olarak İstanbul'un bazı semtleri, adaları, önemli yerleri ve yollarını içeren bir harita kullanılmaktadır.



Şekil 3.51. Örnek çoklu ortam verisi

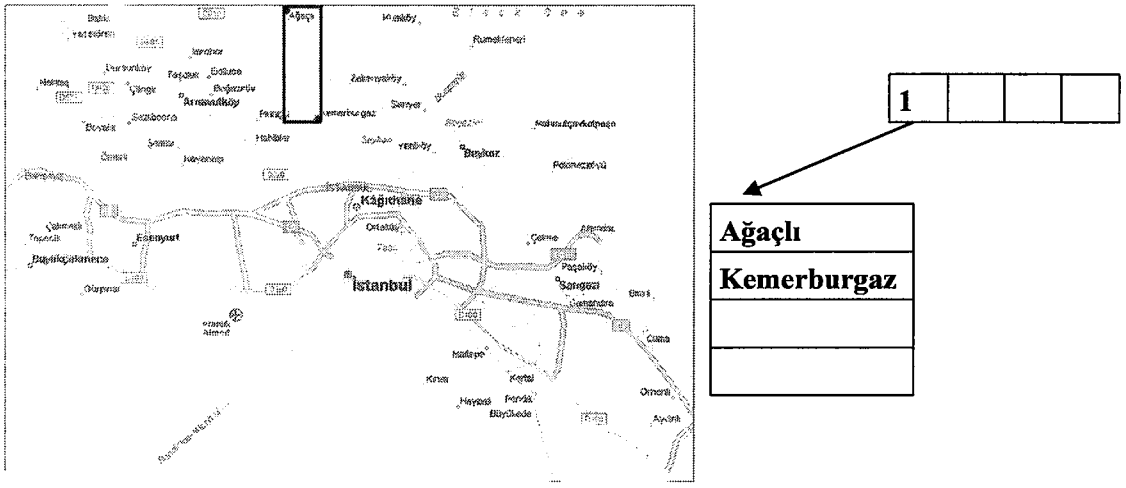
Resimde bulunan noktalar iki boyutlu nokta tipindeki veriyi temsil eder. Arama uzayının başlangıç noktası ($X_0=0, Y_0=0$) ve sınırları (240,165)'dir. Hücre düğümlerinin kayıt kapasitesi $M=4$ ve $m=2$ olarak belirlenmiştir.

1- Ağaçlı(98,164) verisi veri tabanına eklenmeden önce veri tabanı boş durumdadır. Kayıt sayısı $m=2$'den küçük olduğundan dolayı henüz bir bölge oluşmuş durumda değildir.



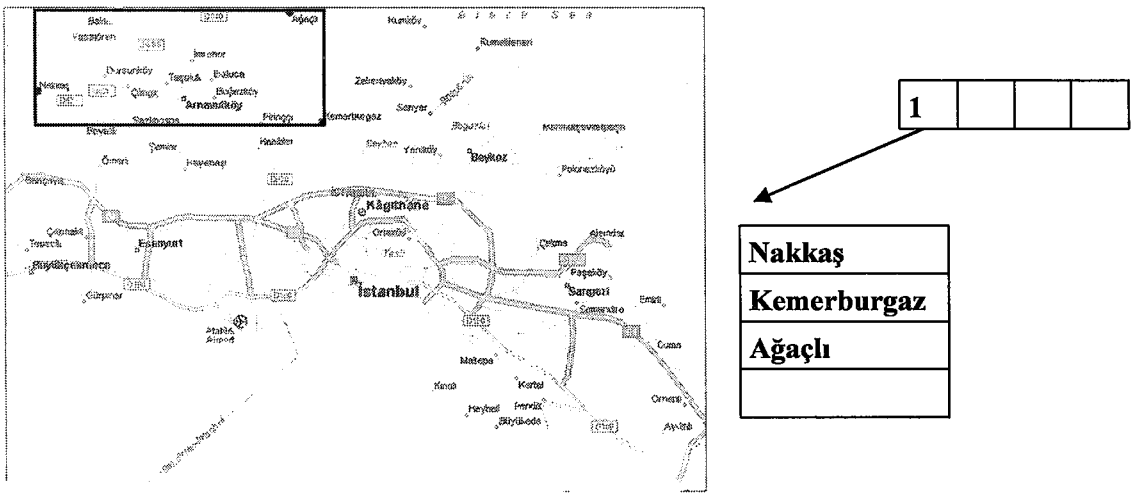
Şekil 3.52. Ağaçlı(98,164) noktası

2- Kemerburgaz(108,126) verisi veri tabanına eklenirken kayıt sayısı *m=2* değerine ulaşır. Bu durumda 1. Bölge ve kök düğüm oluşur. Bölge sınırları ve alt bölge bilgileri kaydedilir.



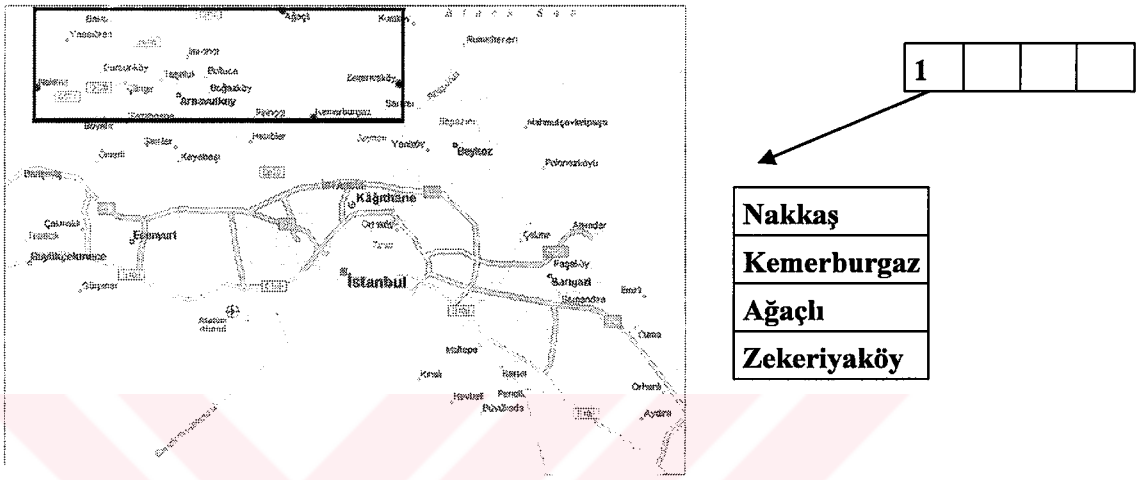
Şekil 3.53. Kemerburgaz(108,126) noktası

3- Nakkaş(10,135) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Bu durumda bu algoritma verinin 1. bölgeye uygun olduğunu belirtmektedir. Veri bu düğüme eklenir. Ardından "ağaç düzenleme" algoritması çalıştırılarak bölge sınırları ve alt bölge bilgileri yenilenir.



Şekil 3.54. Nakkaş(10,135) noktası

4- Zekeriyaköy(140,138) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Bu durumda bu algoritma verinin 1. bölgeye uygun olduğunu belirtmektedir. Veri bu düğüme eklenir. Ardından "ağaç düzenleme" algoritması çalıştırılarak bölge sınırları ve alt bölge bilgileri yenilenir.



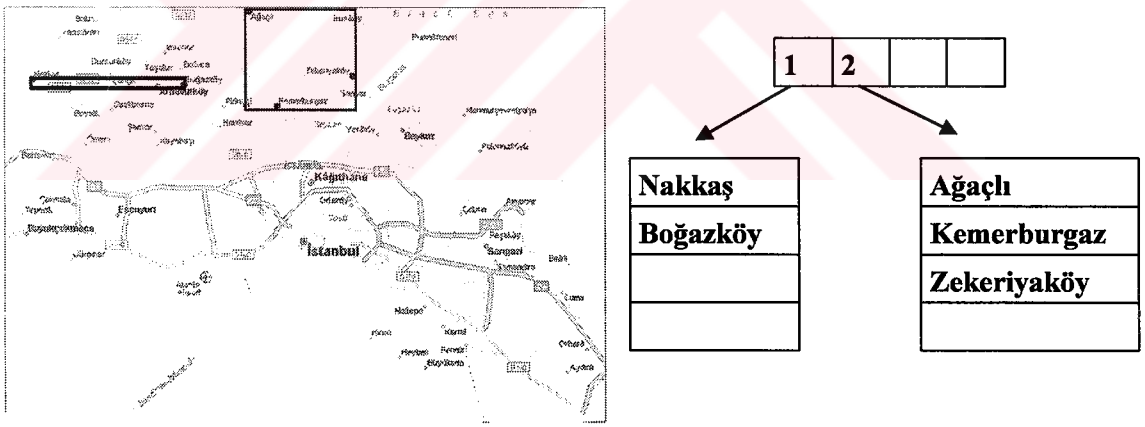
Şekil 3.55. Zekeriyaköy(140,138) noktası

5- Boğazköy(71,135) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Bu durumda bu algoritma verinin 1. bölgeye uygun olduğunu belirtmektedir. Fakat 1 numaralı bölgenin $M=4$ maksimum kayıt sayısına sahip olmasından dolayı "düğüm bölünme" algoritması çağırılır. Bunun sonucunda 2 yeni bölge oluşur. Bu bölgelerden ilkine toplam kapladıkları alan en az olacak şekilde birbirine en yakın 3 kayıt, ikincisine de birbirine en yakın olan 2 kayıt girmektedir.

Çizelge 3.5. R Tree bölünme ihtimalleri ve muhtemel toplam alanlar

A	**B**	**C**	**D**
1.Bölge	*1.Bölge*	*1.Bölge*	*1.Bölge*
Nakkaş (10,135)	Nakkaş (10,135)	Kemerburgaz(108,126)	Kemerburgaz (108,126)
Boğazköy(71,135)	Boğazköy(71,135)	Nakkaş (10,135)	Nakkaş (10,135)
Kapladığı alan : 0	Ağaçlı(98,164)	*Kapladığı alan : 882*	Boğazköy (71,135)
	Kapladığı alan :2552		*Kapladığı alan :882*
2. Bölge		*2. Bölge*	
Ağaçlı(98,164)	*2. Bölge*	Boğazköy (71,135)	*2. Bölge*
Kemerburgaz(108,126)	Kemerburgaz(108,126)	Zekeriyaköy(140,138)	Zekeriyaköy (140,138)
Zekeriyaköy(140,138)	Zekeriyaköy(140,138)	Ağaçlı (98,164)	Ağaçlı (98,164)
Kapladığı alan : 1596	*Kapladığı alan : 2816*	*Kapladığı alan : 2001*	*Kapladığı alan : 1092*
Toplam alan:1596	*Toplam alan:5368*	*Toplam alan:2883*	*Toplam alan:1974*

Muhtemel bölünmelerin kaplayacağı alanlar hesaplanır. Bu tabloya göre A tipi bölünme en az alan kaplayacaktır. Alt bölgeler ve sınırları "ağaç düzenleme" algoritması ile yenilenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.



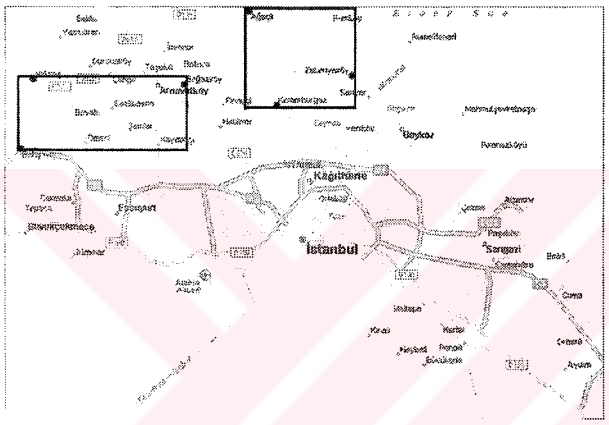
Şekil 3.56. Boğazköy(71,135) noktası

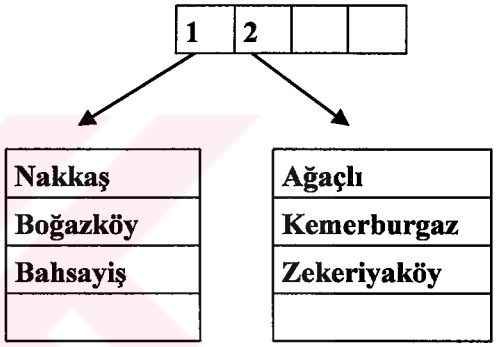
6- Bahsayiş(5,110) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Kök düğümde iki bölge bulunduğundan dolayı verinin bölgelerden tümüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.6. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (71,135)	0	1650	1650
2. bölge	(5,110) – (140,164)	1596	7290	5694

Alan artışının en az olacağı 1. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 1. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.





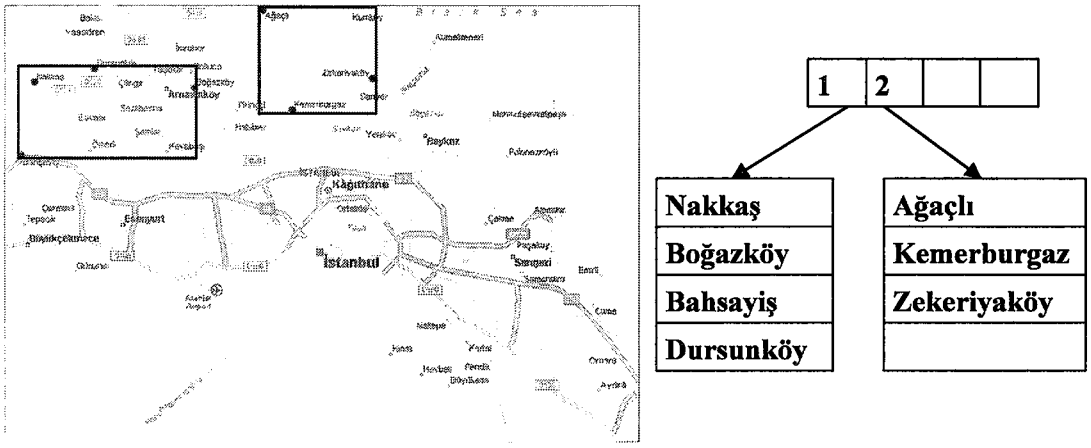
Şekil 3.57. Bahsayiş(5,110) noktası

7- Dursunköy(35,142) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. İki bölge bulunduğundan dolayı verinin bölgelerden her ikisine de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.7. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (71,142)	1650	2112	462
2. bölge	(35,126) – (140,164)	1596	3990	2394

Alan artışının en az olacağı 1. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 1. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.



Şekil 3.58. Dursunköy(35,142) noktası

8- Boyalık(27,125) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. İki bölge bulunduğundan dolayı verinin bölgelerden her ikisine de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.8. R Tree bölge sınırı artışları

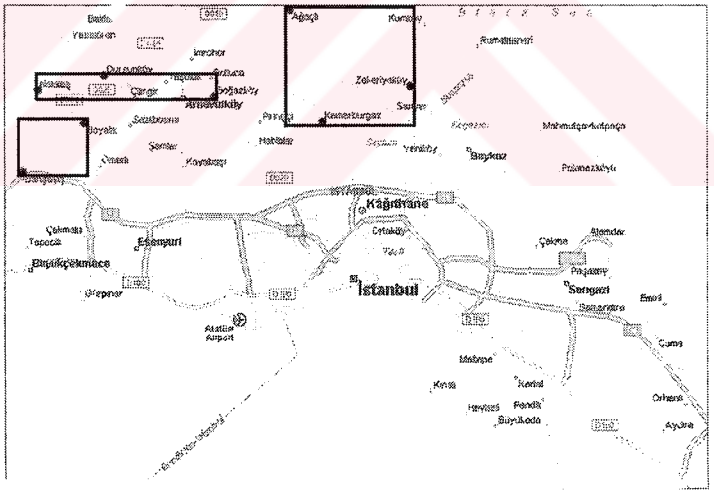
Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (71,142)	1650	2112	462
2. bölge	(35,126) – (140,164)	1596	3990	2394

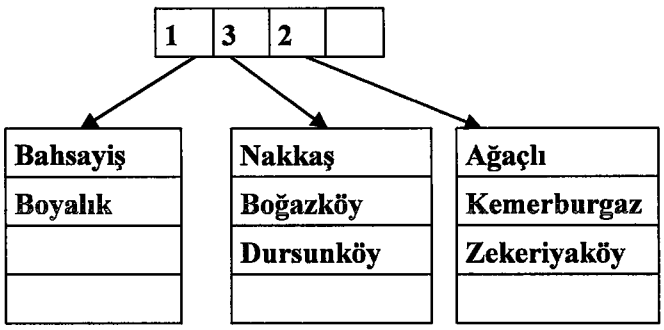
Alan artışının en az olacağı 1. bölge "yaprak seç" algoritması tarafından belirlenir. Fakat 1 numaralı bölgenin $M=4$ maksimum kayıt sayısına sahip olmasından dolayı "düğüm bölünme" algoritması çağırılır. Bunun sonucunda 2 yeni bölge oluşur. Bu bölgelerden ilkine toplam kapladıkları alan en az olacak şekilde birbirine en yakın 3 kayıt, ikincisine de birbirine en yakın olan 2 kayıt girecektir.

Çizelge 3.9. R Tree bölünme ihtimalleri ve muhtemel toplam alanlar

A	**B**	**C**	**D**
1.Bölge	*1.Bölge*	*1.Bölge*	*1.Bölge*
Bahsayiş (5,110)	Bahsayiş (5,110)	Bahsayiş (5,110)	Bahsayiş (5,110)
Nakkaş (10,135)	Nakkaş (10,135)	Boyalık (27,125)	Boyalık (27,125)
Kapladığı alan : 125	Boyalık (27,125)	*Kapladığı alan : 330*	Nakkaş (10,135)
	Kapladığı alan :550		*Kapladığı alan :550*
3. Bölge		*3. Bölge*	
Boyalık (27,125)	*3. Bölge*	Nakkaş (10,135)	*3. Bölge*
Dursunköy (35,,142)	Dursunköy (35,,142)	Boğazköy (71,135)	Boğazköy (71,135)
Boğazköy (71,135)	Boğazköy (71,135)	Dursunköy (35,142)	Dursunköy (35,,142)
Kapladığı alan : 748	*Kapladığı alan : 252*	*Kapladığı alan : 427*	*Kapladığı alan : 252*
Toplam alan:873	*Toplam alan:802*	*Toplam alan:757*	*Toplam alan:802*

Muhtemel bölünmelerin kaplayacağı alanlar hesaplanır. Bu tabloya göre C tipi bölünme en az alan kaplayacaktır. Alt bölgeler ve sınırları "ağaç düzenleme" algoritması ile yenilenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.





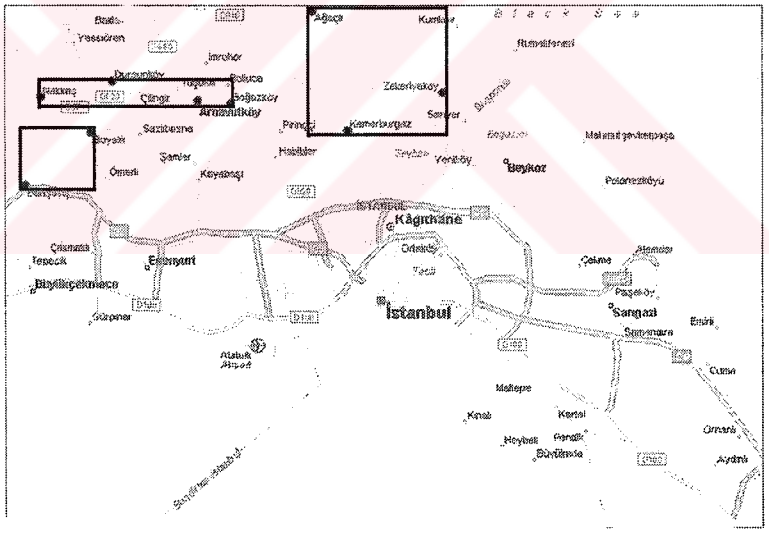
Şekil 3.59. Boyalık(27,125) noktası

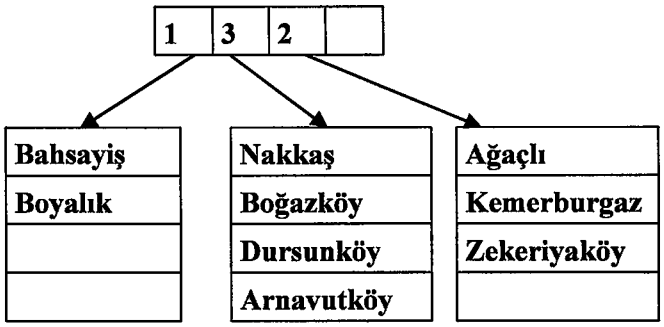
9- Arnavutköy(60,135) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Üç bölge bulunduğundan dolayı verinin bölgelerden her üçüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.10. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (60,135)	330	1375	1045
3. bölge	(10,135) – (71,142)	427	427	0
2. bölge	(60,126) – (140,164)	1596	3040	1444

Alan artışının en az olacağı 3. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 3. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.





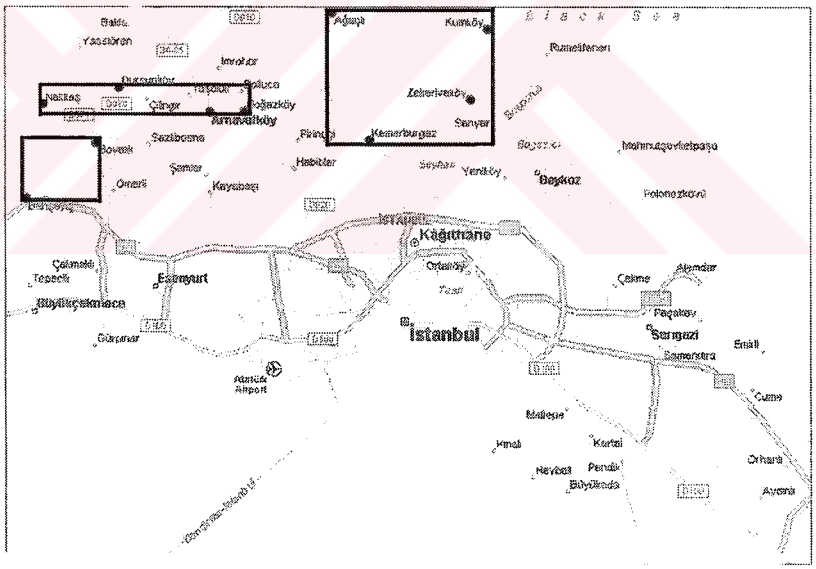
Şekil 3.60. Arnavutköy (60,135) noktası

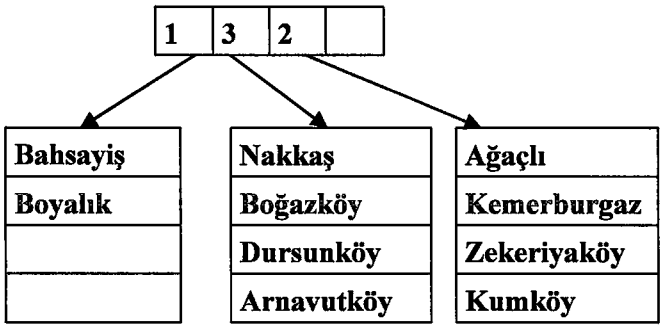
10- Kumköy(144,159) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Kök düğümde üç bölge bulunduğundan dolayı verinin bölgelerden her üçüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.11. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (144,159)	330	6811	6481
3. bölge	(10,135) – (144,159)	427	3216	2789
2. bölge	(98,126) - (144,164)	1596	1748	152

Alan artışının en az olacağı 3. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 2. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.





Şekil 3.61. Kumköy(144,159) noktası

11- Sarıyer(146,130) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Kök düğümde üç bölge bulunduğundan dolayı verinin bölgelerden her üçüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.12. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
1. bölge	(5,110) – (146,130)	330	2820	2490
3. bölge	(10,130) – (146,142)	427	1632	1205
2. bölge	(98,126) – (146,164)	1748	1824	76

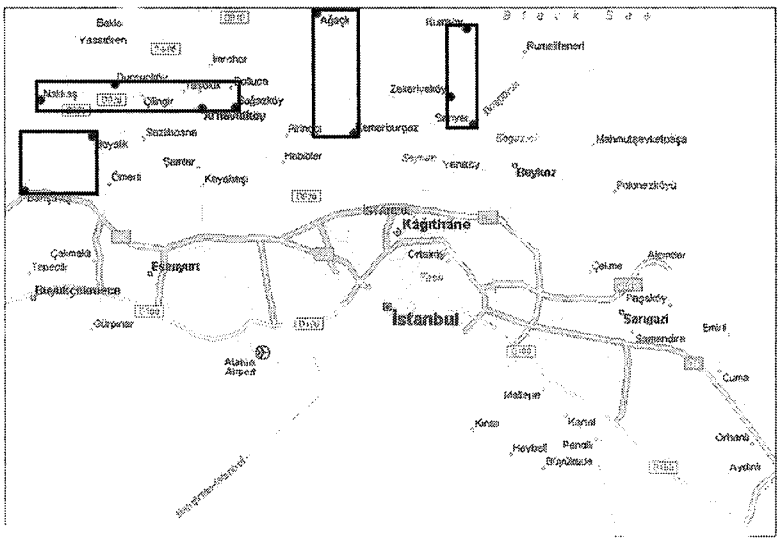
Alan artışının en az olacağı 2. bölge "yaprak seç" algoritması tarafından belirlenir. Fakat 2 numaralı bölgenin *M=4* maksimum kayıt sayısına sahip olmasından dolayı "düğüm bölünme" algoritması çağırılır. Bunun sonucunda 2 yeni bölge oluşur.Bu bölgelerden ilkine toplam kapladıkları alan en az olacak şekilde birbirine en yakın 3 kayıt, ikincisine de birbirine en yakın olan 2 kayıt girecektir.

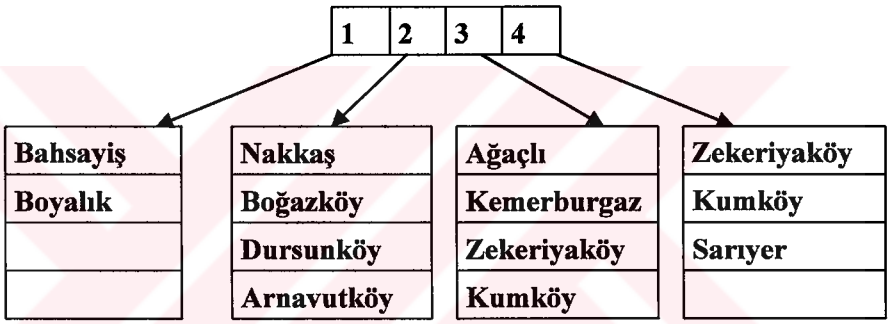
Çizelge 3.13. R Tree bölünme ihtimalleri ve muhtemel toplam alanlar

A	**B**	**C**	**D**
2.Bölge	*2.Bölge*	*2.Bölge*	*2.Bölge*
Ağaçlı (98,164)	Ağaçlı (98,164)	Kemerburgaz (108,126)	Kemerburgaz (108,126)
Kemerburgaz (108,126)	Kemerburgaz (108,126)	Sarıyer (146,130)	Sarıyer (146,130)
Kapladığı alan : 380	Zekeriyaköy (140,138)	*Kapladığı alan : 152*	Zekeriyaköy (140,138)
	Kapladığı alan :1596		*Kapladığı alan :456*
4. Bölge		*4. Bölge*	
Zekeriyaköy (140,138)	*4. Bölge*	Zekeriyaköy (140,138)	*4. Bölge*
Kumköy (144,159)	Kumköy (144,159)	Kumköy (144,159)	Kumköy (144,159)
Sarıyer (146,130)	Sarıyer (146,130)	Ağaçlı (98,164)	Ağaçlı (98,164)
Kapladığı alan : 174	*Kapladığı alan : 58*	*Kapladığı alan : 1196*	*Kapladığı alan : 230*
Toplam alan:554	*Toplam alan:1654*	*Toplam alan:1348*	*Toplam alan:686*

Muhtemel bölünmelerin kaplayacağı alanlar hesaplanır. Bu tabloya göre A tipi bölünme en az alan kaplayacaktır. Alt bölgeler ve sınırları "ağaç düzenleme" algoritması ile yenilenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.

Diğer kayıtların eklenmesi aynı şekilde devam etmektedir.





Şekil 3.62. Sarıyer(146,130) noktası

29- Sarıgazi(194,68) verisi veri tabanına ekleneceği zaman, öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Kök düğümde dört bölge bulunduğundan dolayı verinin bölgelerden her dördüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.14. R Tree bölge sınırı artışları

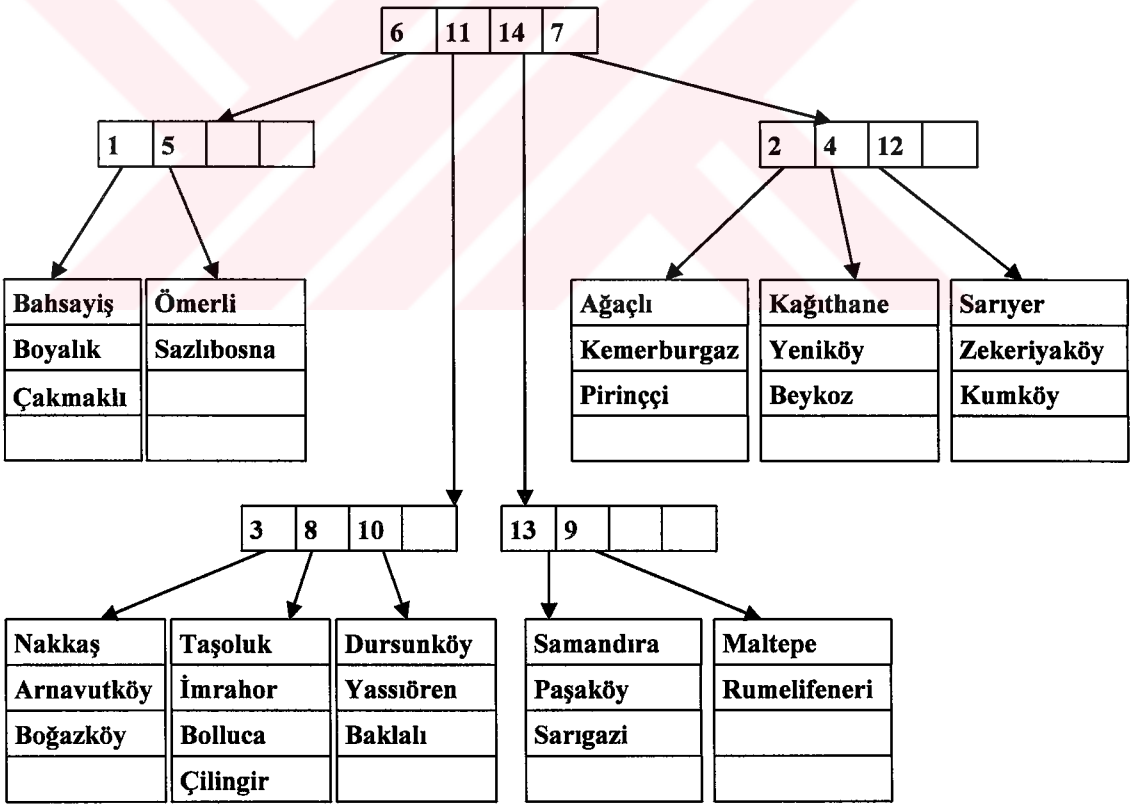
Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
6. bölge	(5,68) – (194,125)	1520		
7. bölge	(88,68) – (194,164)	4828		
11.bölge	(10,68) – (194,164)	1769		
14.bölge	(162,46) – (208,152)	4876	4876	0

Alan artışının en az olacağı 14. bölge "yaprak seç" algoritması tarafından belirlenir. 7. Bölge yaprak düğüm olmadığı için dolayı yaprak düğüme ulaşılıncaya kadar işlem tekrarlanır. Verinin 14. bölgenin alt bölgeleri olan 9 ve 13 bölgelerinden her birisine dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.15. R Tree bölünme ihtimalleri ve muhtemel toplam alanlar

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
9. bölge	(162,46) – (194,152)	530	3392	2862
13. bölge	(194,64) – (208,73)	108	126	18

Alan artışının en az olacağı 13. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 13. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.



Şekil 3.63. Sarıgazi(194,68) noktası

30- Çekme(182,82) verisi veri tabanına eklenirken öncelikle verinin dahil edileceği yaprak düğümü belirleyecek olan "yaprak seç" algoritması çalıştırılır. Kök düğümde dört bölge bulunduğundan dolayı verinin bölgelerden her dördüne de dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.16. R Tree bölge sınırı artışları

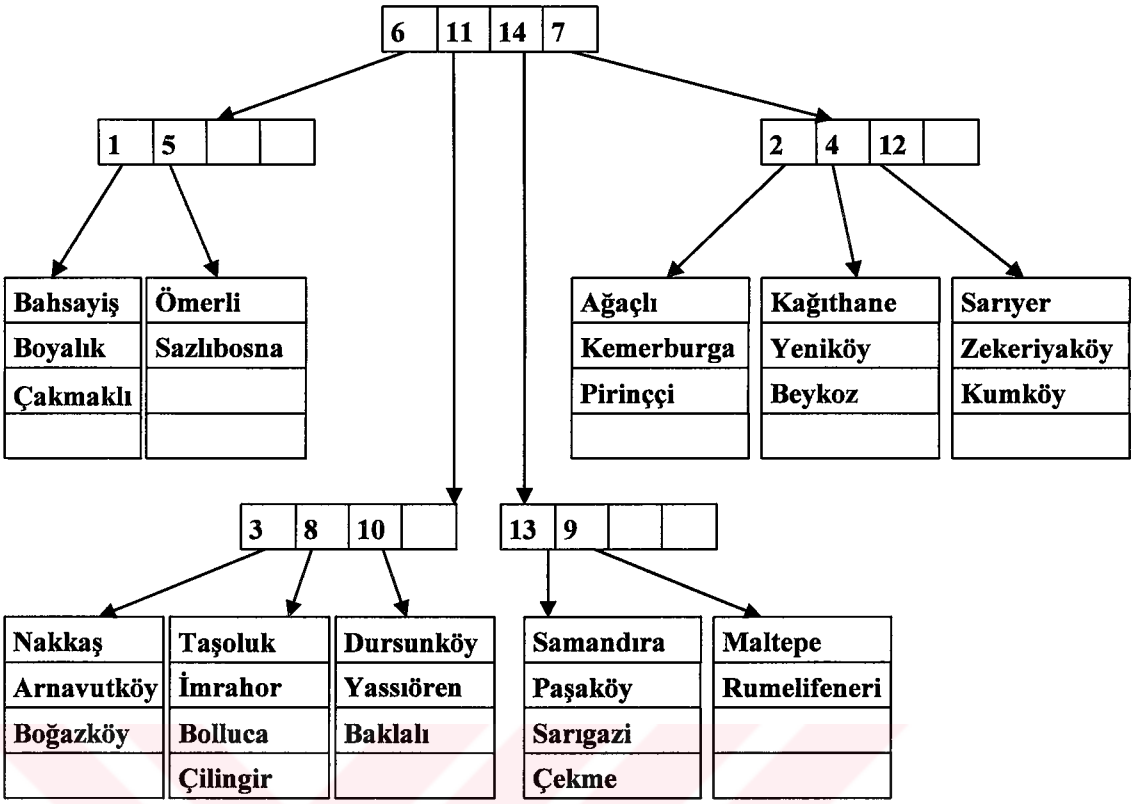
Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
6. bölge	(5,82) – (182,125)	1520		
7. bölge	(88,82) – (182,164)	4828		
11.bölge	(10,82) – (182,164)	1769		
14.bölge	(162,46) – (208,152)	4876	4876	0

Alan artışının en az olacağı 14. bölge "yaprak seç" algoritması tarafından belirlenir. 7. Bölge yaprak düğüm olmadığı için dolayı yaprak düğüme ulaşılıncaya kadar işlem tekrarlanır. Verinin 14. bölgenin alt bölgeleri olan 9 ve 13 bölgelerinden her birisine dahil olduğu taktirde yapacağı alan büyümeleri hesaplanır.

Çizelge 3.17. R Tree bölge sınırı artışları

Bölge Adı	*Bölgenin Yeni Sınırları*	*Eski Alan*	*Yeni Alan*	*Alan Artışı*
9. bölge	(162,46) – (182,152)	530	2120	1590
13. bölge	(182,64) – (208,82)	126	468	342

Alan artışının en az olacağı 13. bölge "yaprak seç" algoritması tarafından belirlenir. Kayıt 13. bölgeye dahil edilir. Bölge sınırları güncellenir. Ağaç yapısı ve bölgeler aşağıdaki gibi olmaktadır.

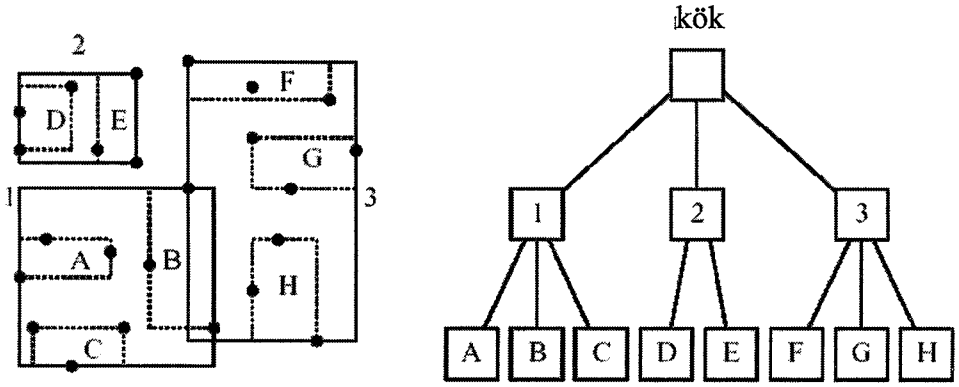


Şekil 3.64. Çekme(182,82) noktası

3.1.2.2. R* Tree

R Tree'nin en başarılı çeşididir. Dikdörtgensel veri için çok boyutlu indeks yapısıdır. Nesneler bir dikdörtgenin köşelerini ifade eden 4 elemanlı bir vektör ile ifade edilmektedirler. R* Tree dikdörtgensel veri setini yükseklik dengeli bir ağaç hiyerarşisi içinde düzenlemektedir. Düğümler ve yapraklar dikdörtgenlere karşılık gelirler ve her biri için bir disk bloğu ayrılmıştır. Bir düğümün dikdörtgeni, çocuklarını en küçük çevreleyen dikdörtgendir. Bu yüzden kök düğümün dikdörtgeni, tüm veri setini çevreleyen en küçük dikdörtgendir. [Beckmann, 1990]

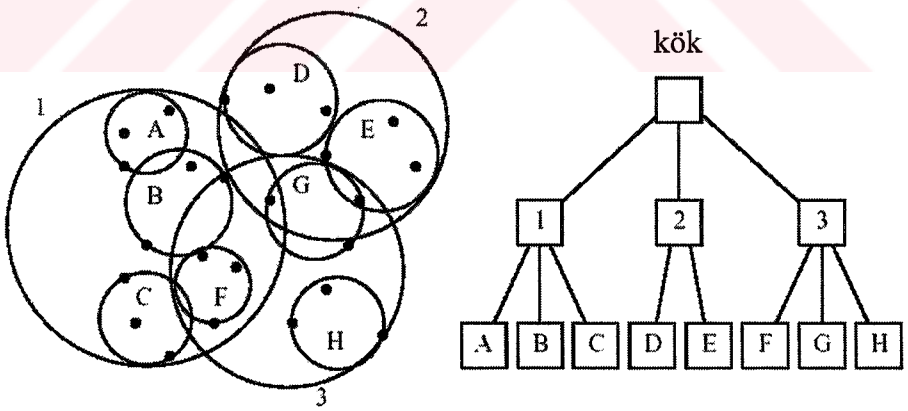
R* Tree, R Tree'nin ekleme ve bölünme algoritmalarında yapmış olduğu değişiklikler ve yeniden ekleme mekanizması sayesinde performans artışı sağlamaktadır.



Şekil 3.65. R* Tree

3.1.2.3. SS Tree

R Tree yapısının bölgeleri oluşturmak için kullandığı dikdörtgen içine alma yönteminin dezavantajlarını gidermek için geliştirilmiştir. SS Tree yapısında veriler dikdörtgenler değil küre seklindeki bölgeler yardımıyla gruplandırılırlar. SS Tree R Tree'nin gelişmiş bir şeklidir. Özellikle yakın komşu sorgulamalarında performans sağlamaktadır.

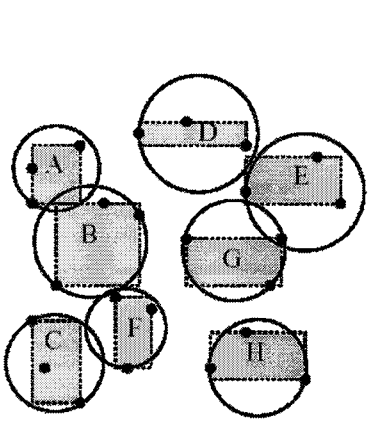


Şekil 3.66. SS Tree

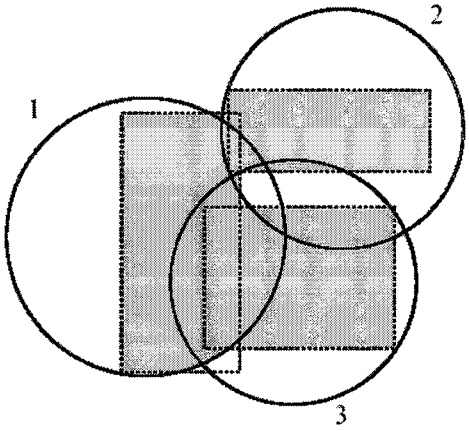
3.1.2.4. SR Tree

R Tree yapısının bölgeleri oluşturmak için kullandığı dikdörtgen içine alma ve SS Tree yapısının bölgeleri oluşturmak için kullandığı küre içine alma yönteminin

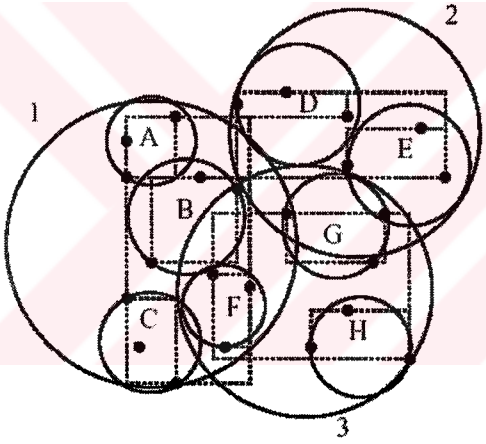
dezavantajlarını gidermek için geliştirilmiştir. SR Tree yapısında düğümler dikdörtgen şeklindeki bölgeler ile veriler küre seklindeki bölgeler yardımıyla gruplandırılırlar. [Katayama, 1997]



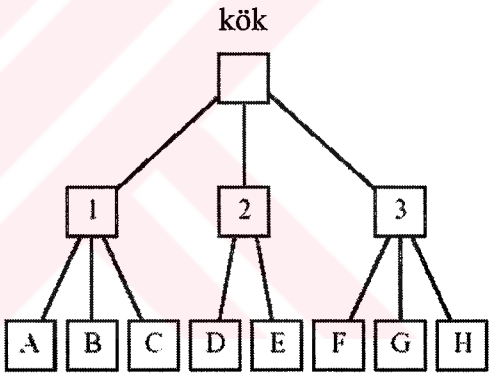
a) Yaprak seviyesi



b) Düğüm seviyesi



c) Verilerin gruplandırılması



d) Ağaç gösterimi

Şekil 3.67. SR Tree

3.1.2.5. Tv Tree

Tv Tree yöntemi, kullanılan veriye dayalı olarak dinamik ve esnek bir şekilde nasıl davranılacağına karar veren bir yöntemdir. Eğer birçok nesnenin bir takım özelikleri birbirine benzer ise, indeks birbiriyle ilişkili bu nesneleri yakın bellek bölgelerinde bulunacak şekilde organize eder. Bu organizasyon Tv Tree yapısının aktif/pasif boyut özellikleri kullanılarak gerçekleştirilir. Eğer indeks vektöründe ağaç yapısına uymayan bir boyut varsa pasifleştirilir. Diğer boyutlar ise aktiftir.

3.1.2.5.1. Tv Tree'de kullanılan terimler

NÇocuk: Tv Tree'nin herhangi bir düğümünde bulunabilen maksimum çocuk sayısı

á: 0 ile k arasında aktif boyut sayısını gösteren bir numaradır.

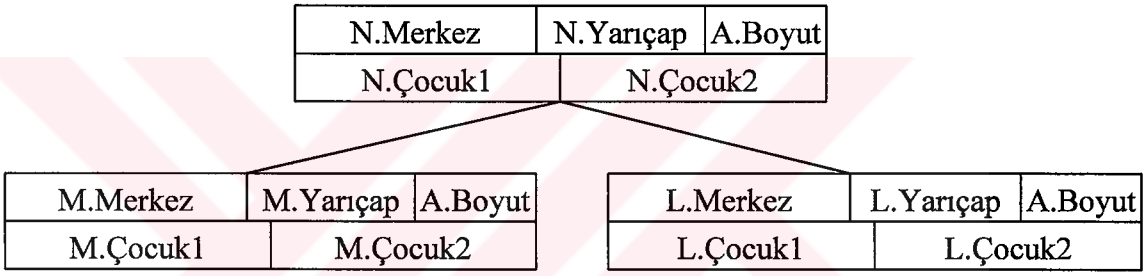
TV(k,NÇocuk,á): k boyutlu *NÇocuk* alt bölgeye sahip ve *á* aktif boyutlu bir TV-Tree yapısını ifade etmektedir.

TV tree'de her bir düğüm bir bölge belirtir. Her N düğümünün 3 alanı vardır ;

N.Merkez: k boyutlu uzayda bölgenin merkezini gösteren bir noktadır

N.Yarıçap: 0'dan büyük bölgenin yarıçapını gösteren bir sayıdır

N.AktifBoyut: 1 ile k arasında bulunan á'nın alt kümesi olan aktif boyutları gösteren bir listedir.



Şekil 3.68. N düğümü tarafından kapsanan bölge

x ve *y*'nin *k* boyutlu veri uzayında ayda birer nokta olduğu taktirde *x* ve *y*'nin arasındaki aktif uzaklık a*u(x,y)* şeklinde gösterilmekte ve şu formülle hesaplanmaktadır:

$$au = \sqrt{\sum x^2{}_i - y^2{}_i}$$

Burada x_i ve y_i *i.* boyuttaki koordinat değerlerini ifade etmektedir. Bu işlemler bir örnek üzerinde gösterildiğinde;

k=200, *á=5* ve *N.AktifBoyut={1,2,3,5,6}* olan bir *N* düğümüne $x=(10,5,11,x_4,13,7,x_7,x_8....x_{200})$ ve $y=(2,4,14,y_4,8,6,y_7,y_8....y_{200})$ noktaları arasındaki aktif uzaklık

$$au(x,y) = \sqrt{(10-2)^2} + (5-4)^2 + (11-14)^2 + (13-8)^2 + (7-6)^2 = 10$$

olmaktadır.Düğüm *N* x_i gibi noktalar içeriyorsa, tüm bu x_i noktalarının *N.Merkez'den* aktif uzaklıkları *N.Yarıçap* değerinden daha büyük olamaz. *N* düğümünde *N.Merkez(10,5,11,13,7,0,0,0,0,....0)* düğüm merkezi ve *N.AktifBoyut={1,2,3,4,5}* ise bu düğümü oluşturan aktif noktaların değerleridir. *N* düğümü aynı zamanda alt düğümlerini gösteren *Çocuk* işaretçisini ve alt düğüm sayısını ifade eden *N.Çocuk* değerini de içerir.

Tv Tree'de tüm veri yaprak düğümlerde tutulur. Tv Tree'deki kök ve yapraklar hariç her bir düğümünün en azından yarısı dolu olmak zorundadır. Örneğin *Çocuk* işaretçilerin en azında yarısı boş olmamaktadır. Eğer *N* bir düğüm ve $N_1....N_r$ de onun çocukları ise $Bölge(N) = U\ Bölge(N_i)$ ile ifade edilmektedir. [Lin, 1994]

3.1.2.5.2. Tv Tree ekleme algoritması

TV Tree Ekleme algoritması verilen kaydın ağaç yapısına dahil edilmesine dayanmakta ve üç adımda gerçekleşmektedir.

1- Yaprak Seçme: TV Tree'ye yeni bir vektör eklerken,eklenen *N.* düğümdeyken (N_i *çocuklu*) kaydın hangi alt düğüme ekleyeceği belirlenir.

2- Bölünme: Belirlenen alt düğümün kapasitesinin dolu olduğu durumda gerçekleşir. Düğüm iki alt düğüme bölünür.

3- Telescobing: Bir N düğümüne yeni bir vektör eklendiğinde aktif boyutlar yeni eklenen veriye uygun olarak yeniden düzenlenmelidir. *Telescobing* bazı aktif boyutların azalmasıyla olabildiği gibi eklenmesiyle de gerçekleşebilir.

3.1.2.5.3. Tv Tree yaprak seçme algoritması

N düğümünün *j* adet alt düğümü olduğu durumda, *v* vektörü için yaprak seçme işlemi için;her bir *j* düğümünün $N_j.Merkez$ değerinden *v* vektörüne olan aktif uzaklıkları hesaplanmaktadır. Aktif uzaklıkların $N_j.Yarıçap$ sınırı içinde kalan değerleri arasında en küçük sonuca sahip olan bölge dallanılacak bölgedir.

Örnek olarak $N_1...N_5$ 'e kadar 5 alt düğüm bulunması durumunda hesaplamalar sonucunda v vektörünün, merkezine en yakın olduğu düğüm seçilmektedir.

3.1.2.5.4. Tv Tree bölünme algoritması

Kapasitesi dolu olan bir *N* yaprak düğümüne bir *v* vektörü eklenmek istenildiğinde düğümün bölünmesi gereklidir. Bölünme aşağıdaki şekilde gerçekleşir;

-*N* düğümünün içine aldığı iki alt düğüm tanımlanır.

-*N* düğümündeki vektörler ikiye bölünür.

-*N* düğümündeki vektörlerin ikiye bölünebildiği tüm olasılıklar incelenir.

Meydan gelen olası alt bölgelerin muhtemel yarıçaplarının en küçük olanı kabul edilerek en iyi bölünme gerçekleştirilmiş olur.

3.1.2.5.5. Tv Tree telescobing algoritması

Vektör *v*'nin *N* düğümüne eklenmesi sırasında *N* düğümünde iki farklı değişiklik meydana gelebilmektedir.

a) *N* düğümünün ikiye bölünmesine neden olabilir.

b) *N* düğümünü aktif boyut kümesi değişebilir.

N düğümü ikiye bölündüğünde düğümün aktif boyutları N_1 ve N_2 alt düğümlerindeki vektör setlerini içine alacak biçimde düzenlenmelidir.

Bölünme gerçekleşmeden, bir ekleme söz konusuysa *N* düğümündeki vektör setine uymayan bir boyut belirlenirse *N* düğümümün aktif boyut kümesinden çıkartılmaktadır.

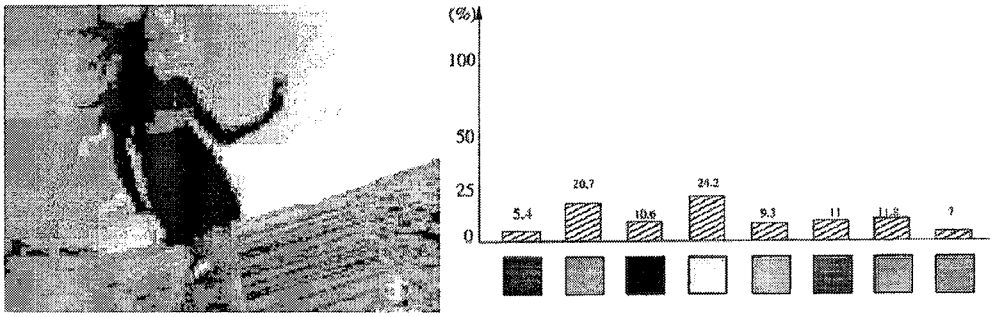
Her iki koşulda da aktif boyut seti kümesinin büyüyüp küçülmesi *telescobing* diye adlandırılır.

3.1.2.5.6. Tv Tree yapısının resim dosyalarında uygulanması

Çoklu ortam veri tabanı sisteminde kullanılan diğer erişim metotları iki ve daha üzeri boyut bilgisi kullanarak verileri indekslemektedir. Bu yöntemler için gerekli parametreleri sağlayan özellik çıkarıcı birim kenar çıkarım ve en küçük çevreleyen dikdörtgen yöntemlerini kullanmaktadır. Halbuki resim histogramı, resmin renk yoğunluğunu gösteren özellik çıkarımı bakımında değerli bir grafiktir. Histogram resmin yoğunluğu ve kontrastı hakkında bilgi verir. Resim histogramı resmin pixel

yoğunlarının basit çubuk grafiğidir. Piksel yoğunlukları x ekseni boyunca, olayların numaraları ise y ekseni boyunca gösterilir.

Özellik çıkarıcı tarafından sağlanan resim histogramı bilgisi, boyut sayısının fazla olması sebebi ile Tv Tree metodu hariç diğer erişim metotları tarafından kullanılamamaktadır.



Şekil 3.69. Örnek resim dosyası ve histogram grafiği

Resimler indekslenirken resim hakkında önemli bilgiler içeren histogram değerini kullanmak benzer resimlerin gruplanması açısından daha uygun olmaktadır. Bu yöntemde resimler özellik çıkarıcı katman tarafından k boyutlu histogram değerlerini gösteren bir vektör şeklinde ifade edilirler. Vektör bilgileri Tv Tree algoritması tarafından değerlendirilerek veriler veri tabanına eklenmektedir.

4. ÖZELLİK ÇIKARICI

Özellik çıkarma işlemi çoklu ortam veri tabanı yapısının en önemli kısımlarındandır. Bu katmanda kullanılan yöntem veri tabanının etkinliğini belirlemektedir ve çoklu ortam verilerinin sınıflandırılmasında kullanılacak, birbirlerine göre ayırt edici özelikleri belirler. Belirlenen bu veriler fiziksel katman tarafından kullanılır. Kullanılan özelliklere örnek olarak; resimler için en küçük çevreleyen dikdörtgen, ses dosyaları için sesin frekansı verilebilir.

Özellik çıkarıcı çoklu ortam verilerinden nesne temelli bilgiyi çıkarmak için kullanılmaktadır. Özellik çıkarım metotları üç kategoriye ayrılır;

A- *Tam Otomatik Çıkarım Metotları:* Çoklu ortam verileri için çıkarım süreci otomatik olarak yapılmaktadır. Herhangi bir kullanıcı yardımı olmaksızın yöntem nesneleri ve özelliklerini belirlemektedir. Bütün çoklu ortam veri tipleri bu yaklaşım tarzı ile yönetilemez. Bu yöntem genellikle ayrılabilir arka plana sahip video karelerinde ve resimlerde kullanılmaktadır.

B- *Yarı Otomatik Çıkarım Metotları:* Bu metotta kullanıcı özellik çıkarım sürecine yardım eder. Metot kullanıcıya özellik çıkarımında sadece yardımcı konumundadır. Kullanıcı nesnelerin özelliklerini çıkarılmasında temel teşkil etmektedir.

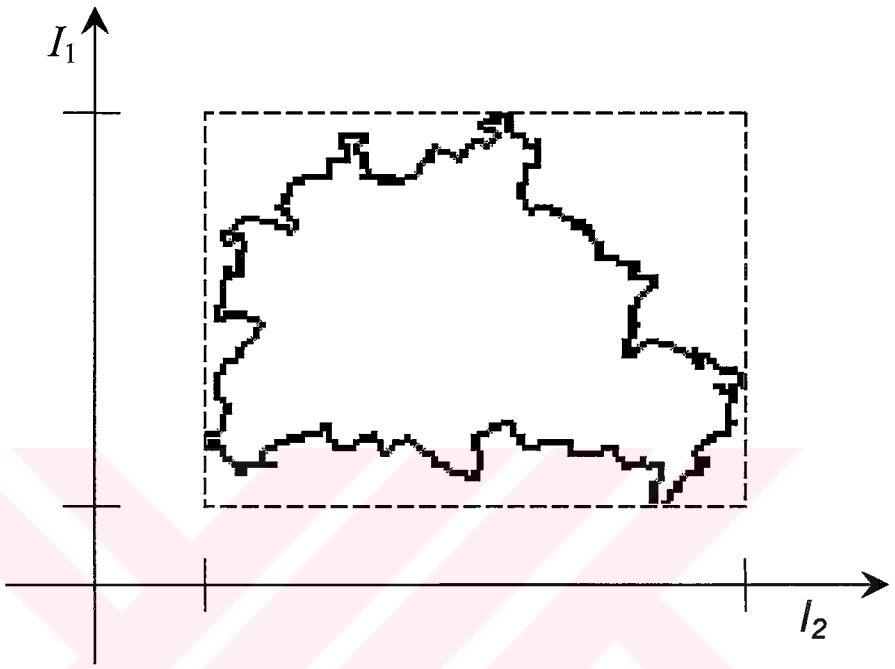
C- *Elle Yapılan Nesne Çıkarımı:* Kullanıcı bilgisayar ilişkisi diğer metotlardan daha fazladır. Çünkü kullanıcı bütün çıkarım sürecini yönetir. Minimum çevreleyen dikdörtgen ve nesne alanını belirlemek için kullanıcının elle yaptığı çizim kullanılmaktadır. Elle yapılan çıkarım çok sıkıcı süreçtir ve çok büyük veri setlerine uygulanamamaktadır

Nesne çıkarımında temel amaç; herhangi tip video veya resim verisini çıkarmadan çıkarım süreci boyunca kullanıcı etkileşimini minimize etmektir. Bu yüzden güçlü bir çıkarıcı yukarıdaki her çıkarım metodu için araçlar içermelidir.

4.1. En Küçük Sınırlayıcı Dikdörtgen

En küçük sınırlayıcı dikdörtgen (MBR); nesne şeklinin tümünü kaplayan en küçük alan veya bölge olarak tanımlanabilir. Çok boyutlu veri kümelerinde kullanılan en basit özellik çıkarım metodudur. En küçük sınırlayıcı dikdörtgen *n* boyutlu uzayda

hedef nesneyi çerçeveleyen geometrik şekildir. Boyut sayısı arttıkça nesneleri temsil eden geometrik şekiller de değişir. Bu alan iki boyutlu veri kümelerinde dikdörtgen veya çemberdir. Üç boyutlu veri kümelerinde ise küp veya küre olmaktadır.



Şekil 4.1. En küçük sınırlayıcı dikdörtgen

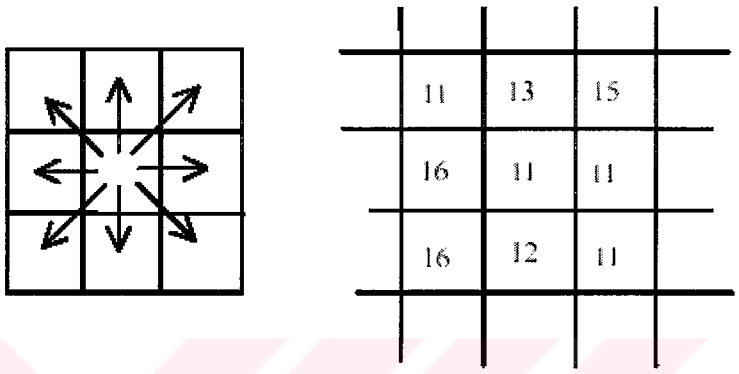
4.2. Kenar Çıkarma

Bir resmin kenarları o resim hakkında çok fazla bilgi tutar. Kenarlar; nesnelerin nerede oldukları, onların şekilleri, büyüklükleri ve özellikleri hakkındaki bilgiyi içermektedir. Bir resmin yoğunluğunun düşük seviyeden yüksek seviyeye çıktığı yer bir kenardır. Kenar çıkarımı için çeşitli özel yöntemler kullanan uygulamalar vardır.

Resim kesimleme; bir resmin oluşumlarını belirlemek için pikselleri alanlara gruplayan resim analizi safhasıdır. Kenar belirleme resim kesimlemenin ilk adımıdır.

4.2.1. Basit kenar çıkarma yöntemi

Basit çıkarıcı, piksel farklarının oluşturduğu serinin maksimum değerine dikkat eder.Ortadaki pikselin değeri etrafındaki 8 piksel değerlerinden sırası ile çıkarılır. Bulunan değerler arasında en büyük olanı seçilir. Bu ortadaki pixelin yeni değerini verir.

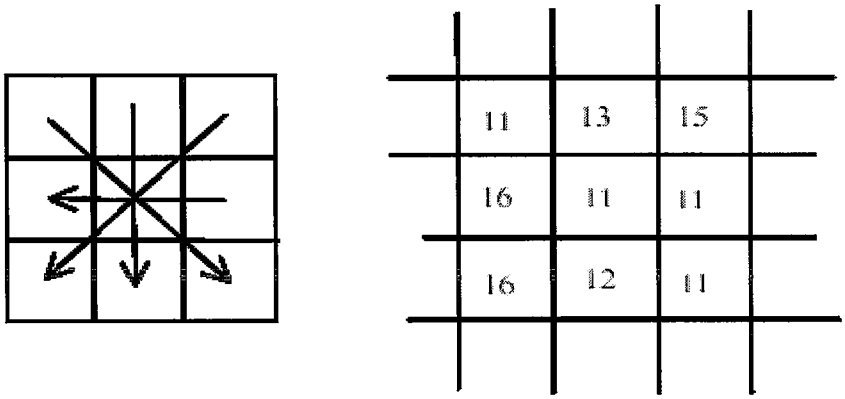


Şekil 4.2. İşlem operatörü ve resim pixel değerleri

Yeni_Piksel=Maksimum{|11-11|,|11-13|,|11-15|,|11-16|,|11-11|,|11-16|,|11-12|,|11-11|}=5

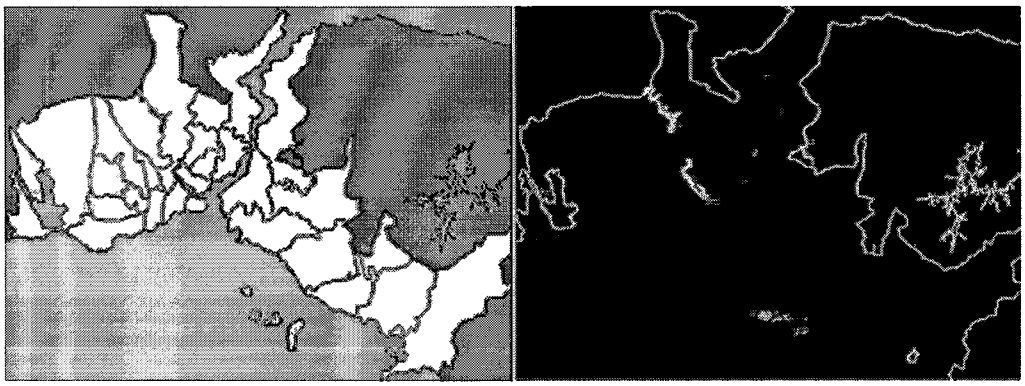
4.2.2. Basit ve hızlı kenar çıkarma yöntemi

Basit ve hızlı kenar çıkarma yöntemi diğer metoda göre daha hızlıdır. Çünkü bu yöntemde her piksel için sadece dört işlem yapmak gerekmektedir.



Şekil 4.3. İşlem operatörü ve resim pixel değerleri

Yeni Piksel=Maksimum{|11-11|,|13-12|,|15-16|,|11-16|}=5



Şekil 4.4. Orjinal resim ve kenarları çıkarılmış resim

4.2.3. Sobel kenar çıkarma yöntemi

Sobel operatörü, resimdeki 2 boyutlu uzaysal eğim ölçülerini temsil eder ve böylece daha çok kenarlara uygun yüksek uzaysal frekans sınırlarını belirtir. Bu yöntemde gri düzey resmini girdi olarak alınır ve bu resmin her noktadaki eğim büyüklüğünü yaklaşık olarak bulunur. Sobel operatörleri köşegen kenarlarda dikey ve yatay kenarlardan daha duyarlıdır.

Sobel operatörü 3*3 lük matrislerden oluşur. *Gy, Gx* 'in 90 derece döndürülmüş halidir. *Gx* ve *Gy* resmin yatay ve düşey olarak kenarlarını bulmak için tasarlanmışlardır. Bunların her ikisi de resme ayrı ayrı uygulanacaksa *Gx* ve *Gy* olarak alınır. Eğer istenirse *Gx* ve *Gy* birlikte de uygulanabilir. Bunun için;

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{hesaplanır.}$$

X yönündeki operatör(Gx)

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

Y yönündeki operatör(Gy)

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Resim Değerleri

0	0	0	0	0	0	2	0	3	3
0	0	0	1	0	0	0	2	4	2
0	0	2	0	2	4	3	3	2	3
0	0	1	3	3	4	3	3	3	3
0	1	0	4	3	3	2	4	3	2
0	0	1	2	3	3	4	4	4	3

Bu resme X ve Y operatörleri sırası ile uygulanır ve ikisinin sonuçlarının mutlak değerleri toplanır. Sonuç Değer=abs(X)+abs(Y)

Sonuç Olarak;

4	6	4	10	14	12	14	4
6	8	10	20	16	12	6	0
4	10	14	10	2	4	2	4
2	12	12	2	2	4	8	8

Örnek olarak ilk 3 pixellik alanı ele alınsın;

0	0	0
0	0	0
0	0	2

X operatörü ile çarpımın sonucu = (-1*0)+(-2*0)+(-1*0)+(0*0)+(0*0)+(0*0)+(1*0) +(2*0) +(1*2)=2

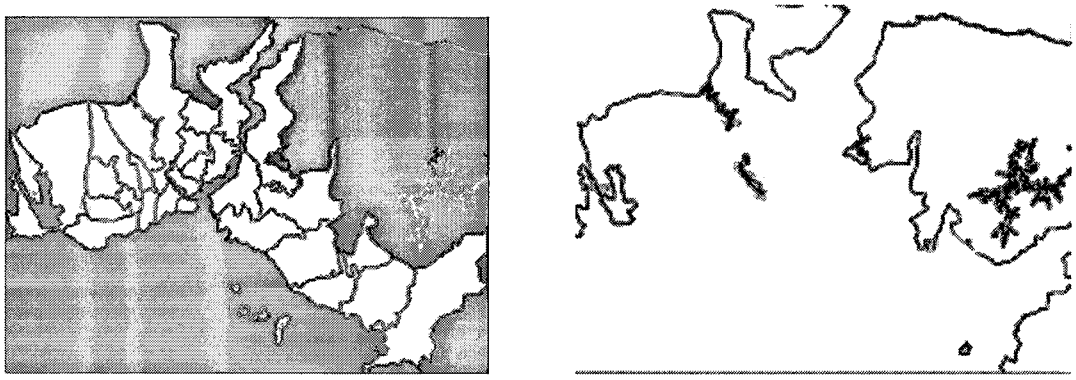
Y operatörü ile çarpımın sonucu = (-1*0)+(0*0)+(1*0)+(-2*0)+(0*0)+(2*0)+(-1*0) +(0*0)+(1*2)=2

Yeni Pixel=abs(x)+abs(y)=2+2=4 sonucu bulunur.

Operatörler üçerlik matris olduğu için resmi üçerlik matrisler şeklinde alırsak satır sonundaki 2 sütün ve sondaki 2 satır pixel işlem yapılmadan kalır. Resim 6*10 matrisi ise 4x8 matrisine dönüşür.Bununla birlikte kenarın yönlendirme açısı bulunur;

$$\phi = \arctan(Gx / Gy)$$

θ açısı sıfır değerini aldığı taktirde maksimum kontrast siyahtan beyaza (solda sağa) doğrudur ve diğer ölçülerde saat yönünün tersi olarak ilerler.



Şekil 4.5. Orijinal resim ve Sobel metoduyla elde edilmiş resim

4.2.4. Roberts kenar çıkarıma yöntemi

Sobel kenar çıkarım yöntemi gibi uygulanır. Diğer masklardan daha küçük etkili alana sahiptir. Kolay etkilenen gürültü değerlerinde kullanılabilir. Maske değerleri;

$$H_r = \begin{matrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} \qquad H_c = \begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Örnek olarak ilk 3 pixellik alanı ele alalım.

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{matrix}$$

X operatörü ile çarpımın sonucu=(0*0)+(0*0)+(-1*0)+(0*0)+(1*0)+ (0*0)+(0*0)+ (0*0)+(0*2)=0

Y operatörü ile çarpımın sonucu=(-1*0)+(0*0)+(0*0)+(0*0)+(1*0)+ (0*0)+(0*0)+ (0*0)+(2*0)=0

Yeni Pixel=abs(x)+abs(y)=0+0=0 sonucu bulunur.

4.2.5. Prewit kenar çıkarıma yöntemi

Diğer yöntemler gibi uygulanmaktadır. Dikey ve yatay kenarlarda köşegen kenarlardan daha iyidir. Maske değerleri;

$$H_r = \begin{matrix} -1 & -1 & -1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \qquad H_c = \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$

Örnek olarak ilk 3 pixellik alanı ele alalım.

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{matrix}$$

X operatörü ile çarpımın sonucu=(-1*0)+(-1*0)+(-1*0)+(0*0)+(0*0)+(0*0)+(1*0)+(1*0)+(1*2)=2

Y operatörü ile çarpımın sonucu=(1*0)+(0*0)+(-1*0)+(1*0)+(0*0)+(-1*0)+(1*0)+(0*2)+(-1*2)=-2

Yeni Pixel=abs(x)+abs(y)=2+2=4 sonucu bulunur.

4.2.6. Frei-Chen kenar çıkarma yöntemi

Diğer yöntemler gibi uygulanmaktadır. Maske değerleri;

$$H_r = \begin{matrix} 0 & 0 & -1 \\ \sqrt{2} & 0 & \sqrt{2} \\ 0 & 0 & -1 \end{matrix} \qquad H_c = \begin{matrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{matrix}$$

Örnek olarak ilk 3 pixellik alanı ele alalım.

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{matrix}$$

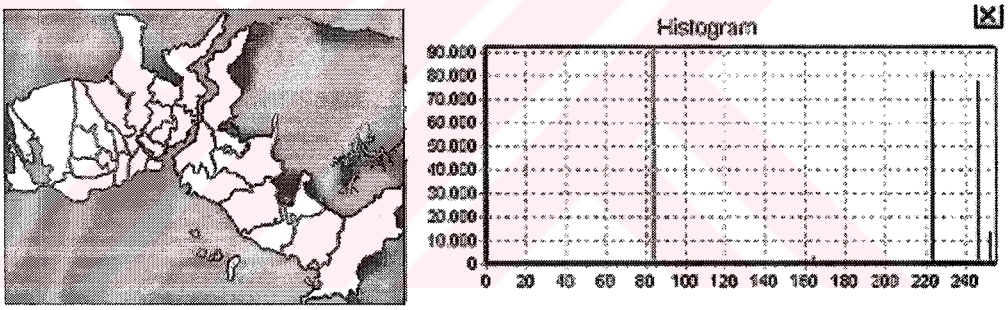
X operatörü ile çarpımın sonucu=(0*0)+(0*0)+(-1*0)+($\sqrt{2}$ *0)+(0*0)+($\sqrt{2}$ *0)+ (0*0) +(0*0)+(-1*2)=-2

Y operatörü ile çarpımın sonucu=(-1*0)+ (-$\sqrt{2}$ *0)+(-1*0)+(0*0)+(0*0)+(0*0)+(1*0) +($\sqrt{2}$ *0)+ (1*2)=2

Yeni Pixel=abs(x)+abs(y)=2+2=4 sonucu bulunur.

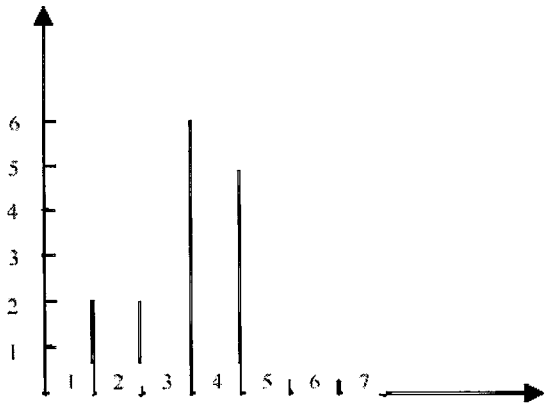
4.3. Histogram

Resim histogramı, resmin renk yoğunluğunu gösteren özellik çıkarımı bakımında değerli bir grafiktir. Histogram resmin yoğunluğu ve kontrastı hakkında bilgi verir. Resim histogramı resmin pixel yoğunlarının basit çubuk grafiğidir. Piksel yoğunlukları x ekseni boyunca, olayların numaraları ise y ekseni boyunca gösterilir.



a) Resim b) Pixel yoğunlukları

4	4	3	3
4	4	3	3
4	1	2	3
0	1	2	3



c) Resim d) Pixel Yoğunlukları

Şekil 4.6. Basit Histogram gösterimleri

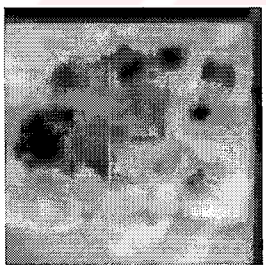
Koyu renk resimlerin pixel dağılımları sol, parlak renkli resimlerin pixel dağılımları sağ yoğunluklu olmaktadır. İdeal bir resimde dağılımın tek biçimde olması gerekmektedir.

5. GÖRSEL SORGU ARA YÜZÜ

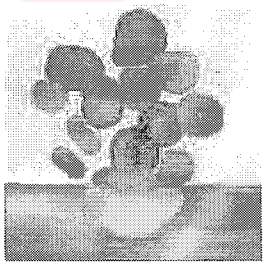
Kullanıcıdan alınan uzaysal parametrelere bağlı olarak veri tabanını sorgulayarak geri dönen sonuçları kullanıcıya yansıtan birimdir. Bu katmanda grafik arabirim kullanılmaktadır. Grafik arabirim kullanıcıdan aldığı sorgu nesnesine karşılık gelen cevabı kullanıcıya iletir. Sorgulama sisteminde sorguları oluşturmak için grafiksel sorgu ara yüzü kullanılmaktadır. Bu ara yüz uzaysal ve yörünge sorgu tipleri için dizayn edilmiş sorgu belirleme pencerelerinden oluşmaktadır. Bu pencereler kullanılarak istenilen sayıda uzaysal ve yörünge sorgular oluşturulabilir. İki farklı sorgulama biçimi bulunmaktadır.

***A-** Örnek Temelli Sorgular:* Verilen örnek çoklu ortam verisine benzer tüm veriler kullanıcıya iletilir. Örnek veriden elde edilen özellik vektörü ile daha önceden veri tabanına eklenmiş ve indekslenmiş verilere ait özellik vektörleri karşılaştırılır.

***B-** Özellik Vektörü Temelli Sorgular:* Kullanıcı aradığı verinin özellik vektörü parametrelerini (renk, doku, şekil, histogram, frekans gibi) girerek buna karşılık gelen verileri elde eder. Örnek olarak bir resim veritabanı için, kullanıcı görsel sorgu ara yüzünde eliyle çizdiği bir resme en çok benzeyen resimleri sonuç olarak sistemden ister.



a) Örnek temelli sorgu



b) Özellik vektörü temelli sorgu

c) Sistemin cevabı

Şekil 5.1. Sorgular ve sistemin cevapları

Her iki tip sorgulama görsel sorgu ara yüzü kullanıcıdan aldığı uzaysal sorgu nesnesini özellik çıkarıcı birime iletir. Bu birimden alınan bilgi doğrultusunda veri tabanında aranan nesneye karşılık gelen muhtemel sonuçlar belirlenir. Bu sonuçlar kullanıcıya iletilerek son karşılaştırmanın kullanıcı tarafından yapılması istenir.

Şekil 5.2.'de CBIRD çoklu ortam veri tabanı sisteminin görsel sorgu ara yüzü görülmektedir.

Şekil 5.2. CBIRD çoklu ortam veri tabanı sistemi sorgulama ara yüzü

6.SONUÇ

Uzaysal erişim metotlarının tümünün ortak noktası uzaysal parametrelere sahip nesnelerin özelliklerini kullanarak disk üzerine yerleştirilmesi ve sorgulanmasında etkinlik kazandırmasıdır. Bilindiği üzere yığın, kuyruk gibi ana ve yardımcı bellek erişim metotları doğrusal özelliktedir. Bir verinin belleğin hangi noktasında bulunduğuna dair geliştirilmiş algoritmalar da tek anahtar değer üzerinde çalışmaktadır. Örneğin indeks sıralı dosyalar tek anahtar değere göre disk izlerinde bulunan verilerin en büyük değerli olanlarını bir üst seviye indekste tutarak yönetmektedirler. Hash yöntemleri tek anahtar değer için üreteç fonksiyonlar kullanmaktadır. Benzer olarak ağaç yapıları, yakın büyüklüğe sahip verileri ortak alt dallara, tek anahtar değer kullanarak yerleştirmektedir.

Klasik yöntemler olarak gruplandırılan doğrusal erişim metotları, çoklu ortam verilerini yönetmek için kullanılamazlar. Bilgisayarların yapısal kısıtlamalarından dolayı verinin en alt düzeyde ifade edilmesinde doğrusal erişim metotları kullanılmak zorundadır. Bu yüzden geliştirilmiş veya bundan sonra geliştirilecek olan uzaysal erişim metotları iki alt katmandan oluşmak zorundadır. İlk katman uzaysal özelliklere sahip nesnelerin bu özelliklerini değerlendirmesi ve bilgisayarın yapısal özelliklerine uygun hale getirilmesi işleminden sorumludur. İkinci katmanda doğrusal hale getirilmiş ham verinin disk üzerine yerleştirilmesi ile ilgili yöntem belirlenir.

Çok boyutlu verilerin bilgisayar ortamında saklanmasındaki problemlerden birisi bilgisayar sistemlerinin tek boyutlu olan yardımcı ve ana bellek yapıları ve algoritmaların bu yapılara uygun hale getirilmesidir. Her ne kadar uzaysal erişim metotları nesnelerin disk üzerinde saklanacakları yerleri kesin çizgilerle belirleyebilse de tek boyutlu B+ ağaç yapısı kadar etkin olamayacaktır.

Tek boyutlu B+ ağaç yapısında, benzer ve yakın anahtar değerlere sahip veriler disk üzerinde aynı disk sayfalarında saklanmaktadır. Bu, sorgulama algoritmalarının etkinliğini arttırmakta ve kayda erişim sürelerini azaltmaktadır. Özellikle aralık sorgulamalarında çoğu kez diğer alt ağaçlara (dolayısıyla disk sayfalarına) bakılmadan sonuca ulaşılabilmektedir.

Çok boyutlu verilerin indekslenmesinde en etkin çözüm boyut sayısını azaltmaktır. Yüksek boyutlarda benzerlik sorgulamaları boyut faktöründen dolayı büyük

bir problemdir. Genel olarak boyut azaltılması işleminde; iki boyutlu uzayda, birbirine yakın olan nesnelerin tek boyutlu uzayda birbirine yakın olmaları istenmektedir.

Yukarıdaki kriterler dikkate alındığında uzaysal erişim metotlarının etkinlikleri incelenirken yakın ve/veya benzer uzaysal özelliklere sahip nesnelerin aynı yada birbirini takip eden bellek bölgelerinde olmaları göz önünde bulundurulmalıdır. Verileri yerleştirme süresi göz önüne alındığında etkinliği düşük bir algoritma, aynı yada birbirini takip eden bellek bölgelerinde, yakın ve benzer uzaysal özelliklere sahip nesneleri saklayarak gruplandırabiliyorsa, diğer algoritmaların aksine erişim ve arama süreleri bakımından daha etkindir.

Uzay kontrollü erişim metotlarından sabit ızgara ve ızgara dosyası, veri uzayına eşit dağılmış nesneler söz konusu olduğunda basit ve hızlı yöntemlerdir. Fakat belli noktalarda verilerin yoğunlaşması söz konusu olduğunda yöntemler avantajını kaybetmektedirler. Ayrıca bu yöntemlerin verilere erişimi çok basit şekilde gerçekleşmektedir.

Bunu ortadan kaldırmak için Dörtlü ağaç yöntemi geliştirilmiştir. Dörtlü ağaç yöntemi diğer uzay bölümlemeli yapılardan farklı olarak bölünen hücreleri dizin yapısından daha etkin olan ağaç yapısına dönüştüren bir yaklaşımdır.

Dörtlü ağaç yönteminde de diğer tüm ağaç yöntemlerinde olduğu gibi bazı dar boğazlar mevcuttur. Ağaç yapısının dengesiz olması yüzünden, Dörtlü ağaç yapısı bazı durumlarda erişim süresi bakımından etkinliğini kaybetmektedir. Ayrıca bölünmenin dinamik olarak gerçekleşmesi veri ekleme sırasında zaman kaybına yol açmaktadır.

Çoklu boyutlu veriyi indekslemek için kullanılan tüm bu yöntemler bazı konularda yetersiz kalmışlardır. Çok boyutlu verinin tek boyuta indirgenmesinde daha hızlı, basit ve etkin bir yöntem ihtiyacı eğri yöntemlerini ön plana çıkarmaktadır. Eğri yöntemleri veri uzayını sabit çözünürlükteki hücrelere bölmekte ve bu hücrelere birbirini izleyen numaralar vermektedir. Birbirini izleyen numaralara sahip hücreler bellek üzerinde birbirini takip eden bölgelere kaydedilmektedir. Eğri yöntemlerinden Z sıralaması'nın engellenemez bazı "atlama"ları bulunmaktadır. Bu atlamaları azaltmak için Hilbert eğrisi ve Gray kodu yöntemleri geliştirilmiştir. Tüm boşluk dolduran eğri yöntemleri aktif boyut sayısını azaltmaya yöneliktir. Yapılan çalışmalarda uzay kontrollü yöntemler arasında eğri yöntemlerinin hızlı, basit ve etkin olduğu gözlenmiştir.

Veri kontrollü yapılar verilerin tüm uzaysal özelliklerini kullanarak verileri sınıflandırırlar. Veri kontrollü yapı yöntemlerinin birçoğu R Tree yöntemini temel kabul etmektedir. Bu yöntemler yükseklik dengeli ve bir dönüşüm yönteminden çok gerçek bir uzaysal erişim metodu olmasından dolayı tercih edilmektedir.

Yapılan bu çalışmada elde edilen sonuçlar, uzaysal nesnelerin veri kontrollü yapılar tarafından çok boyutlu bir ağaç yapısıyla indekslenmesinin, dönüşümler yoluyla tek boyuta indirgenerek geleneksel erişim metotlarıyla indekslenmesinden daha uygun olacağını göstermektedir.

KAYNAKLAR

Beckmann N., Hans-Peter Kriegel,1990, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". ACM SIGMOD International Conference on the Management of Data, Atlantic City

Bentley J. L., Jerome H. Friedman,1979, "Data Structures for Range Searching", Computing Surveys

Bially T., 1969, "Space-filling curves: Their generation and their application to bandwidth reduction", IEEE Trans. on Information Theory

Butz Arthur R., 1969, "Convergence with Hilbert's space filling curve", Journal of Computer and System Sciences

Butz Arthur R., 1971, "Alternative Algorithm for Hilbert's Space-Filling Curve", IEEE Transactions on Computers

David J. Abel, David M. Mark, 1990, "A comparative analysis of some two-dimensional orderings", Int. J. Geographical Information Systems

Dyer R., 1982, "The space efficiency of quadtrees", Computer Graphics and Image Processing

Faloutsos C., 1986, "Multiattribute hashing using gray codes", Proc. ACM SIGMOD

Faloutsos C.,1992, "Analytical results on the quadtree decomposition of arbitrary rectangles", Pattern Recognition Letters

Freeston M.,1987, "The BANG File: A New Kind of Grid File", Association for Computing Machinery Special Interest Group on Management of Data Annual Conference

Gaede V., 1998, Oliver Gunther, "Multidimensional Access Methods", ACM Computing Surveys

Guttman, A., 1984, "R-Trees, A Dynamic Index Structure for Spatial Searching", ACM SIGMOD, Boston, Mass.

Jagadish H.V., 1990, "Linear clustering of objects with multiple attributes" ACM SIGMOD

Jagadish H.V., 1997, "Analysis of the Hilbert curve for representing two-dimensional Space", Information Processing Letters

Katayama N., S. Satoh, 1997, "The SR-tree: An Index Structure for High Dimensional Nearest Neighbor Queries", ACM SIGMOD

Lawder J., 1999, "The Application of Space-Filling Curves to the Storage and Retrieval of Multi-dimensional Data", Birkbeck College, University of London

Lempel A., J. Ziv, 1984, "Compression of two-dimensional images", NATO ASI Series

Lin, K.-I., H. V. Jagadish,1994, "The TV-tree - An Index Structure for High-dimensional Data.", VLDB Journal 3: 517-542.

Moon B., Jagadish H.V., Faloutsos C., Saltz J. H.,1996, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve", Technical Report CS-TR-3611 / UMIACS-TR-96-20, University of Maryland

Orenstein J., 1986, "Spatial query processing in an object-oriented database system", Proc. ACM SIGMOD

Orenstein J. A., F.A. Manola. PROBE, 1988, "Spatial Data Modeling and Query Processing in an Image Database Application", IEEE Transactions on Software Engineering

Pagel, B., May 1993, "Towards an Analysis of Range Query Performance" ,CM SIGACT-SIGMOD-SIGART Symposium, Washington, D.C.

Patrick Edward A., Douglas R. Anderson, F. K. Bechtel, 1968, "Mapping multidimensional space to one dimension for computer output display", IEEE Transactions on Computers

Robinson, J. T. 1981, "The k-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes", ACM SIGMOD.

Rong Y., Christos Faloutsos,1991, "Analysis of the clustering property of Peano curves" Techn. Report CS-TR-2792, UMIACS-TR-91-151, Univ. of Maryland

Roussopoulos, N., S. Kelley, May 1995, "Nearest Neighbor Queries", ACM-SIGMOD, San Jose, CA.

Sellis T., Nick Roussopoulos, ChristosFaloutsos, 1987, "The R+Tree: A Dynamic Index for Multi-Dimensional Objects", 13th VLDB Conference, Brighton

Shaffer C.A.,1988, "A formula for computing the number of quadtree node fragments created by a shift", Pattern Recognition Letters

ÖZGEÇMİŞ

30.04.1980 yılında Kırklareli'nde doğdum. İlk öğrenimimi Babaeski/Kırklareli'de, orta eğitimimi Kırklareli Anadolu Lisesi'nde, lise eğitimimi Edirne Fen Lisesinde tamamladım. 1998-2002 yılları arasında Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde öğrenim gördüm. 2002 yılında Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisansa başladım. Aynı yıl Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi olarak çalışmaya başladım. Halen aynı görevi sürdürmekteyim.