# KONUMSAL ERİŞİM YÖNTEMLERİ
(SPATIAL ACCESS METHODS)

**Hayri SEVER[1], Haşmet GÜRÇAY[2], Serdar AK[3]**
[1]Çankaya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Balgat, Ankara
[2]Hacettepe Üniversitesi, Matematik Bölümü, Beytepe, Ankara
[3]NETCAD, Cyberplaza B-409, Bilkent, Ankara
sever@cankaya.edu.tr

## ÖZET

Otomatik konumsal veri toplama araçlarının kullanımının hızlıca artmasına paralel olarak konumsal içeriklerinin hacim olarak genişlemesi, geleneksel yöntemlerin ötesinde konumsal veritabanlarının etkin kullanılabilmesini gündeme getirmiştir. Konumsal verilere etkin erişim için, hızlı ve doğru sonuçlar üreten dizinleme yapılarının gerekliliği kaçınılmazdır. Konumsal erişim yöntemleri bilgisayar bilimlerinin çekirdek konularından birisini teşkil ettiğinden dolayı, yeni yöntemler veya yöntem eniyileştirmeleri devamlı önerilmekte ve birçok alanda kendisine uygulama bulmaktadır. Yeni yöntem önermede, değişmeyen ister konumsal erişim yöntemlerinin zaman ve uzay karmaşıklığının belli seviyelerde tutulmasıdır. Bu derleme makalede tüm bu gereksinimler ve bu gereksinimler sonucunda ortaya çıkmış bulunan bazı dizinleme yapıları ve yöntemleri incelenmektedir.

**Anahtar Kelime:** Konumsal erişim yöntemleri, konumsal veritabanları, dizinleme yöntemleri.

## ABSTRACT

Due to the fact that expansion of spatial content volume in parallel with rapidly increase in the use of spatial data acquisition and capturing tools has caused to bring effective use of spatial databases beyond conventional methods to the attention of database community. In order to provide effectively access to spatial data, multidimensional indexing structures become a de facto requirement. Since spatial accessing methods are of core subjects of computer science, new methods or extensions have been continously proposed and pave themselves the way for many application areas. Despite of many new things about these methods there are some static requirements about them such as plausable time and space complexity. In this survey article all these requirements and some interesting and valuable indexing structures and methods around them are explored.

**Key Words:** Spatial access methods, spatial databases, indexing methods.

## 1.GİRİŞ

Yaygın uzaktan algılama uygulamaları ve otomatik veri toplama gereçleri sayesinde konumsal veriler olağanüstü boyutlara ulaşmakta ve bunlar konumsal veritabanlarında saklanmaktadırlar. Geleneksel veri yapısı ve erişim gereçleri bu tip

verilerin içsel olarak barındırdıkları konumsal bilgileri yorumlamakta çeşitli zorluklarla karşılaşmaktadır.

Konumsal sorguları verimli bir şekilde gerçekleştirebilmek için adına *dizin* diyeceğimiz bir veri yapısına dayalı özel erişim yöntemlerine ihtiyaç bulunmaktadır. Bu erişim yöntemleri bir sorgunun gerçekleştirilmesi sırasında bakılması gereken nesnelerin kümesini daraltarak sorgunun hızlanmasını sağlamaktadır (Rigaux, vd.,2002).

Coğrafyada, meteorolojide, astronomide ve geometride yeni uygulama alanları konumsal erişime gerek duymakta ve "en yakın komşu" sorgusu gibi bir sorgu türünü kullanmak isteyebilmektedirler. Bu sebeple eğer uygulama alanı uzayında yakınlık önemliyse konumsal erişim yöntemleri için veriler disk sayfalarında bu yakınlığa uygun olarak öbeklendirilmelidirler (Salzberg, 1996).

Nokta ve pencere sorgularını konumsal sorgulara örnek olarak gösterebiliriz. Bu çeşit sorgularda bir noktayı kapsayan ya da bir diktdörtgen ile kesişen geometrilere sahip nesneler aranmaktadır. Konumsal birleşim ise, bir diğer önemli sorgu örneği teşkil etmektedir. Bu durumda, iki ayrı nesneler kümesinden bazı konumsal ilişkileri sağlayan nesne çiftleri birleştirilerek elde edilir. Eğer geometrileri kesişiyorsa bir özelliğe sahip bir nesne başka bir özelliğe sahip başka bir nesneyle birleştirilebilir. Elde edilen, her iki özelliğe de sahip yeni bir nesnedir (Rigaux, vd., 2002).

İki konumsal nesne üzerinde geometrik hesaplamaların yapılması hafıza ve merkezi işlemci biriminin kullanımı bakımından pahalı bir işlem olduğundan, üzerinde bu tür işlemlerin gerçekleştirilmesi gereken nesneleri etkili bir veri erişim yöntemi ile en aza indirmek önemlidir (Rigaux, vd.,2002). Sözü geçen konumsal erişim yöntemleri 3. Kesimde inceleyeceğiz.

## 2. KONUMSAL VERİNİN ORGANİZASYONU

Konumsal veritabanları ile ilgili gereksinimleri daha iyi anlayabilmek için önce konumsal verinin bazı temel özelliklerine bakmalıyız ve anlamalıyız.

### a. Konumsal Verilerin Özellikleri

Konumsal veriler oldukça karmaşık bir yapıya sahiptir. Konumsal bir nesne tek bir noktadan oluşabileceği gibi uzayda gelişi güzel dağılmış olan binlerce poligondan meydana geliyor olabilir. Konumsal veri genellikle devingen yapıdadır. Eklemeler ve silmeleri güncellemeler takip etmektedir ve bu bağlamda kullanılmakta olan veri yapıları zamanla bozulmaya yer vermeyecek şekilde bu devingen davranışı destekliyor olmalıdır. Konumsal veri çok büyük hacimli olabilmektedir. Örnek olarak birkaç gigabyte bellek alanı kaplayan coğrafi haritaları verebiliriz.

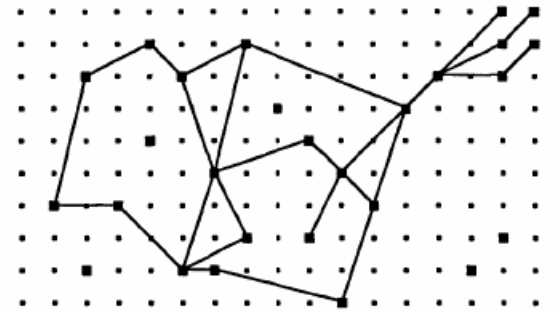
Geometrik nesneleri sabit boyutlu ilişkisel veritabanı tablolarında tutmak genellikle mümkün olmamaktadır; onun yerine Açık Yer-konumsal Konsorsiyum (Open Geospatial Consortium: OGC) tarafından iyi tanımlanmış metin standartı önerildiğini

dikkatinize getirmek isteriz (OGC, 2003). Bir geometrik nesne için ne kadar veri tutulacağı önceden kesin olacak bilinememektedir. Bu nedenle klasik bir veri tabanı tablosu oluşturup verileri onun içine atma yaklaşımı konumsal veriler için uygun değildir. Bununla birlikte, geometrik nesneleri tek bir tabloda tutmak ve böylece ilişkisel modelin matamatiksel formal zenginliğini ve basitliğini kullanmak kaygısı ile, OGC iyi bilinen metin yaklaşımını önermiştir. İyi bilinen metin, geometrik nesnelerin Tablo 1'deki gibi (örneğin, Ankara İl Sınırı, Tuz Gölü'nün Topolojik Şekli, apartmanların koordinatları, haritada bir nokta, vb.) tek bir yalın katar şeklinde tutulmasını sağlamaktadır.

Tablo 1. İyi bilinen metin kullanan konumsal bir veritabanı tablosu örneği.

| GID | Xmin | Ymin | Xmax | Ymax | WKT_Geometry |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 50 | 60 | LINESTRING(0 0, 10 10, 20 25, 50 60) |
| 1 | 15 | 20 | 15 | 20 | POINT(15 20) |

Konumsal verilerin işlenebilmesi için ilişkisel cebire (ya da onu kapsayan ve Yapısallaştırılmış Sorgulama Diline (Structural Query Language: SQL) diline temel teşkil eden ilişkisel çoklu analizi modeline) tamamlayıcı olan ROSE (Robust Spatial Extensions) cebiri önerilmiştir (Guting and Schneider, 1995). ROSE cebiri, geometrik işlemlerde analitik hesaplama ile bilgisayar aritmetiği arasındaki uyumsuzluğu gidermek için geliştirilmiş *realm* (Guting and Schneider, 1993) konumsal veri alanını (ya da kümesini) esas almıştır. realm'ler tüm geometrik nesneleri bir ızgaraya oturtan ve veri değerlerinin bu ızgara noktaları dışında bir nokta olmasını engelleyen bir temel konumsal uzaydır (Şekil 1). ROSE cebirinin konumsal bir sorgulama dilinin matematiksel modeli olarak önerildiğini not etmek isteriz. ROSE cebiri geometrik nesneler arasındaki olası ilişkileri tanımlayan 4-kesişim modeline (Egenhofer & Franzosa, 1991) indirgenebilir.



Şekil 1. Bir realm örneği.

Egenhofer ve meslektaşları tarafından tanımlanan 4-kesişim modeli, iki geometrik nesnenin iç ve sınır değerlerini olası ilişkileri tanımlamak için kullanır. Bir A kümesinin içini $A^o$ ve sınırını $\partial A$ göstermek üzere iki $A_1$ ve $A_2$ geometrik nesnesi için dört farklı ilişki kümesi bulunmaktadır. Bu durumda da karşımıza Tablo 2' de görüldüğü gibi $2^4$=16 değişik sonuç ortaya çıkmaktadır.

Tablo 2. 4 Kesişim Modelindeki 16 değişik olasılık.

| $A_1 \cap \partial A_2$ | $\partial A_1 \cap A^\circ_2$ | $A^\circ_1 \cap \partial A_2$ | $A^\circ_1 \cap A^\circ_2$ | İlişki |
|---|---|---|---|---|
| Ø | Ø | Ø | Ø | $A_1$ ve $A_2$ ayrık |
| Ø | Ø | Ø | ≠Ø | |
| Ø | Ø | ≠Ø | Ø | |
| Ø | Ø | ≠Ø | ≠Ø | $A_2$, $A_1$ 'in içinde |
| Ø | ≠Ø | Ø | Ø | |
| Ø | ≠Ø | Ø | ≠Ø | $A_1$, $A_2$ nin içinde |
| Ø | ≠Ø | ≠Ø | Ø | |
| Ø | ≠Ø | ≠Ø | ≠Ø | |
| ≠Ø | Ø | Ø | Ø | $A_1$, $A_2$ ye değer |
| ≠Ø | Ø | Ø | ≠Ø | $A_1$ ve $A_2$ eşit |
| ≠Ø | Ø | ≠Ø | Ø | |
| ≠Ø | Ø | ≠Ø | ≠Ø | $A_1$, $A_2$ 'yi kapsar |
| ≠Ø | ≠Ø | Ø | Ø | |
| ≠Ø | ≠Ø | Ø | ≠Ø | $A_2$, $A_1$ 'yi kapsar |
| ≠Ø | ≠Ø | ≠Ø | Ø | |
| ≠Ø | ≠Ø | ≠Ø | ≠Ø | $A_1$, $A_2$ 'yi örter |

Tablo 2' de verilen olasılıkların sekiz tanesi geçersiz, iki tanesi de simetriktir. Bu sebeple geriye altı ayrı ilişki kalmaktadır ve bunlar *ayrık, içinde, değer, eşit, kapsar ve örter* olarak adlandırılmaktadır.

Yukarıda tanımlanan topolojik işlemler, konumsal veritabanı üzerinde ifade edilebilen türde sorgu işlemlerini oluşturur. Topolojik ilişkileri işleme ön koşulunu, noktasal (verilen bir sorgu penceresince kapsanan noktalar kümesi) ya da bölgesel (verilen bir sorgu penceresi ile kesişen geometrik nesneler) arama işlemlerinin doğru ve hızlı bir şekilde gerçekleştirilmesi oluşturur. Bu gibi arama işlemlerini desteklemek için çok boyutlu özel erişim yöntemlerine ihtiyaç duyulmaktadır. Böylesi yöntemlerin tasarımındaki asıl problem uzayda yakın bulunan nesnelerin tam bir sıralamasının yapılmasının mümkün olmamasıdır. Diğer bir deyişle iki ya da daha çok boyutlu bir uzaydan tek boyutlu uzaya, daha yüksek boyutlu konumsal uzayda birbirlerine yakın bulunan nesnelerin tek boyutlu sıralanmış bir dizide birbirlerine yakın bulunmalarını sağlayacak şekilde bir dönüşüm bulunmamaktadır. Bu da çok sayıda iyi anlaşılmış ve etkili erişim metodlarına sahip olan geleneksel veritabanlarına nazaran konumsal alanda etkili erişim yöntemlerinin tasarlanmasını çok daha zor hale getirmektedir (Gaede, vd.,1998).

Çok boyutlu erişim yöntemlerinin, konumsal veriye ve uygulamalarına bağlı olarak sağlaması gerekli bir takım gereksinimler bulunmaktadır (Gaede, vd.,1998):

(1) Devingenlik: Veri nesneleri herhangi bir sıra ile çıkarıldıklarında ve eklendiklerinde erişim yöntemleri değişimleri devamlı surette takip etmelidirler.

(2) İkincil ve üçüncül depolama yönetimi: Kapasitesi devamlı artmakta olan ana belleklere rağmen tüm veritabanını hafızada tutmak çoğunlukla mümkün olamamaktadır. Bu nedenle erişim yöntemlerinin ikincil ve üçüncül saklama birimleri ile bütünleştirilmeleri gerekmektedir.

(3) Geniş yelpazede desteklenen işlemler: Erişim yöntemleri başka işlemlerden (silme gibi) feragat ederek sadece bir tür (çağırma gibi) işlemi desteklemekle sınırlı kalmamalıdır.

(4) Giriş verisiyle giriş sırasının bağımsız olması: Erişim yöntemleri, girilen veri düzensiz olduğunda ya da giriş sırası değiştiğinde verimliliklerini korumalıdırlar. Bu nokta özellikle veri farklı boyutlarda farklı şekillerde dağılım göstermekteyse bilhassa önemlidir.

(5) Ölçeklendirme: Erişim yöntemleri veritabanındaki büyümelere iyi uyum göstermelidirler.

(6) Zaman verimliliği: Konumsal aramalar hızlı olmalıdır. Tasarım amaçlarından önemli bir tanesi tek boyutlu dengeli ağaçların (B-Ağaçları) performans karakteristiklerini yakalayabilmektir; Erişim yöntemleri veri giriş sırasına bağlı olmaksızın her tür veri dağılımı için en kötü durumda logaritmik arama performansı gösterebilmelidir.

(7) Boyut verimliliği: Verinin büyüklüğü ile kıyaslandığında bir dizin boyut olarak daha küçük olmalı ve böylelikle belirli bir depolama kullanışlılığını sağlayabilmelidir.

(8) Eş zamanlılık ve güvenlilik: Modern veritabanlarında çok sayıda kullanıcı eş zamanlı olarak güncelleme, çağırma ve ekleme yaptıklarında, erişim yöntemleri belirgin performans kayıplarına yol açmaksızın güvenli işlem yönetimi sağlamalıdır.

(9) En az etki: Mevcut bir veritabanına bir erişim yönteminin entegrasyonu sistemin mevcut parçaları üzerinde en az etkiye sebep olmalıdır.

**b. Tanımlar ve Sorgu Türleri**

Fiziksel uzayla ilgili olarak iki yaygın model bulunmaktadır: uzaya-dayalı ve nesneye-dayalı model. Veritabanı perspektifinden bakıldığında nesneye-dayalı modeller en iyi seçenek olmaktadırlar. Nesneye-dayalı modeller fiziksel uzayı ayrık, ayırt edilebilen ve konumsal olarak referanslanmış varlıklardan oluşuyormuş gibi ele almaktadır. Geometrik özellikleri bir veri modeline bütünleştirebilmek için verileri geometri türünde en az bir alanı bulunan nesneler olarak tanımlamak gerekli

olmaktadır. Daha açık ifade etmek gerekirse denilebilir ki, veri modeli olağan basit veya bileşik veri türlerinin yanı sıra geometrik veri türlerinin de desteklemelidir (Clementini, vd., 2000).

Konumsal veritabanı nedir? Genel olarak kabul görmüş bir tanım olmamakla birlikte konumsal veritabanı için aşağıdaki özellikler verilebilir (Güting, 1994):

- Veri modelinde konumsal veri türlerini ve konumsal sorgu dilini sunmaktadır.

- Gerçekleştirimi aşamasında konumsal veri türlerini desteklemektedir ve en azından konumsal dizinlemeyi ve konumsal birleşim için etkili algoritmaları sağlamaktadır.

Konumsal veritabanlarında arama yapmak için kullanılmakta olan çok boyutlu erişim yöntemleri nokta erişim yöntemleri (NEY) ve konumsal erişim yöntemleri (KEY) olarak ikiye ayrılmaktadır. Nokta erişim yöntemleri özellikle nokta (sadece noktaları saklayan) veritabanlarında konumsal aramalar yapmak için geliştirilmişlerdir. Noktalar iki ya da daha çok boyutlu uzaylarda bulunuyor olsalar da herhangi bir genişletme oluşturmamaktadırlar. KEY'ler ise aksine çizgi, poligon ve hatta çok boyutlu polihedron gibi genişletilmiş nesneleri yönetebilmektedir. Literatürde genellikle konumsal erişim yöntemleri ile çok boyutlu erişim yöntemleri eşanlamlı olarak yer almaktadır. Benzer amaçlarla kullanılmakta olan diğer bir terim ise konumsal dizin ya da konumsal dizin yapısıdır. Verilen nesnelerin genel olarak d-boyutlu bir Öklid uzayında ($E_d$) ya da bunun bir alt uzayında bulunduğu kabul edilmektedir. Bu uzaydan evren ya da esas uzay olarak da bahsedilebilmektedir. Konumsal bir veritabanında saklanmakta olan her nesnenin tek bir konumu bulunmaktadır ve bu konum nesnenin *d* adet koordinatı ile ifade edilmektedir (Gaede, vd.,1998).

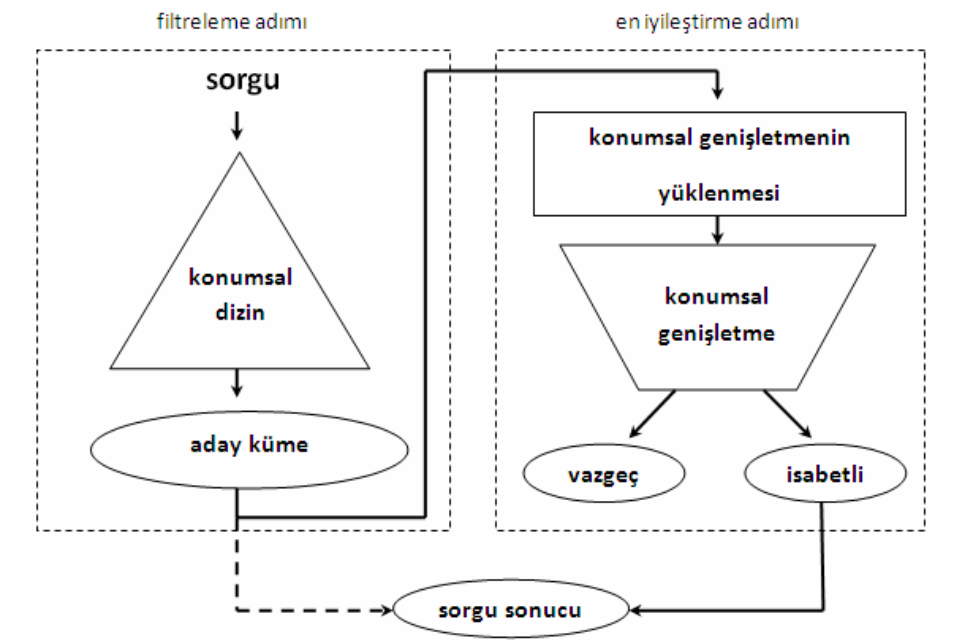
Konumsal veritabanlarında aynı zamanda hem konumsal niteleyicileri ve hemde konumsal olmayan niteleyicileri de içeren birleşik nesneler de yer alabilmektedir. Konumsal veritabanı literatüründe *geometri, şekil* ve *konumsal uzantı* terimleri *konumsal genişletme* ile eşanlamlı olarak kullanıla gelmektedir. (Gaede, vd.,1998)

Kesişme, komşuluk ve kapsama gibi konumsal işlemler geleneksel seçme ve birleştirme işlemlerine göre çok daha zaman alıcıdırlar. Bunun sebebi konumsal nesnelerin şekillerindeki düzensizliklerden kaynaklanmaktadır. Örneğin, iki polihedronun kesişimini alındığında her iki nesnenin tüm noktalarının test edilmesi gerekliliğinin yanı sıra, sonuçta ortaya çıkan nesnenin daima polihedron olma zorunluluğu bulunmamaktadır (Ooi, vd., 2001).

Konumsal veritabanı uygulamalarında nesne tanımlamaları ve dizin yapıları ayrı ayrı ele alınmaktadır. Karmaşık yapılara sahip olan nesneler erişimle ilgili sebeplerden dolayı daha basit yapılara yakınsanmaktadır. Bu yakınsamayı yapmanın sebebi karmaşık konumsal nesneler üzerinde daha pahalı testleri gerçekleştirmeden önce uygun olmayan nesneleri mümkün olduğu kadar çok elemektir. Daha sonra kalan nesneler ya doğrudan doğal uzaylarında dizinlenmekte ya da parametrik bir uzaya dönüşümleri yapıldıktan sonra dizinlenmektedirler (Ooi, vd., 2001).

Dizinler aynı boyuttaki basit kayıtlar söz konusu olduğunda daha elverişli çalışmaktadırlar. Bu sebepten ötürü konumsal bir nesne dizine girilmeden önce asıl şeklinden soyutlanmaktadır. Bu orjinal veri nesnesine bir yaklaşım olarak alınabilecek ve orjinal verinin, orjinal veri nesnesini içeren sınırlayıcı dikdörtgen ya da küre gibi basit bir şekle indirgenmesi ile yapılabilir. En küçük sınırlayıcı dikdörtgen ise, ilgili nesneyi kapsayan En Küçük Sınırlayıcı Dikdörtgen (Minimal Bounding Box: MBB) olarak tanımlanır ve bu bir konumsal verinin konumsal genişletmesi olarak kullanılır (Gaede, vd.,1998).

Bir dizin bir nesnenin nesne referansı (ya da *objectID*) ile birlikte o nesnenin MBB 'nı tutuyor olabilir. Bu yaklaşımda dizin bir dizi aday çözümleri sunabilmektedir. Sonraki filtreleme aşamasında dizinden elde edilen her bir aday çözümün MBB'sinin istenilen özellikleri sağlamasının nesnenin gerçek geometrisinin de bu özellikleri sağlamasını garanti edip etmediğine bakılır. Eğer durum böyleyse aday çözüm doğrudan çözümler kümesine atılabilir, ancak MBB'nin bunu kanıtlamak için yeterli kalmadığı durumlar da olmaktadır. Bu sebeple bir en iyileştirme aşaması ile aday nesnelerin gerçek geometrik bilgilerinin ikincil hafıza alanından çağrılması ve bunlar ile gerekli testlerin yapılması gerekli olmaktadır. Eğer bu aşamada sorgu sonucu doğru çıkarsa nesne çözüm kümesine aktarılmaktadır (Şekil 2).




Şekil 2. Konumsal sorgu süreci.

Basitleştirilmiş bir dizin yaratmanın diğer bir yolu ise, her bir kaydın şeklini daha basit (sınırlı sayıda köşe noktasına sahip dışbükey poligonlar gibi) geometrik nesnelerin birleşimi olarak ifade etmekdir. Bu yaklaşım şekline *ayrıştırma* denilir.

Bir dizinin kapladığı en az hafıza alanına dizinin uzay verimliliği denilmektedir. Dizinin zaman verimliliğinden de bahsedilebilir. Kullanıcının hissetiği geçen zaman önemlidir ancak bu tür ölçümler yapıldığında sonuçların gerçekleştirme ayrıntıları, donanım kullanımı ve diğer dış etkenlere bağlı bulunduğu unutulmamalıdır. Bu

bakımdan literatürde daha objektif bir performans ölçüsüne sıkça rastlanılmaktadır: Bunlardan biri arama sırasında meydana gelen disk erişimlerinin sayısıdır. B-Ağacı ile beraber popüler hale gelmiş bulunan bu yaklaşımdaki temel kabul, aramaların çoğunun CPU'dan çok disk (I/O) giriş/çıkış'ına bağlı olduğudur. Ne varki bu yaklaşım her zaman doğru olmamaktadır. Nesnelerin karmaşık şekillere sahip olduğu uygulamalarda iyileştirme adımı CPU'yu yoğun olarak kullanabilmekte ve CPU-I/O dengesinin yönünü değiştirebilmektedir. Yine de bir tasarım objektifi olarak disk erişimlerinin sayısını en aza indirmeye çalışmak doğru bir yaklaşım olmaktadır. Bununla beraber pratikte işletim zamanın nasıl ve ne şartlar altında harcandığına dikkat edilmelidir (Gaede, vd.,1998).

İlişkisel veritabanlarının oldukça popüler olmasının en önemli etmenini altında yatan formal matematiktir (ilişkisel cebir, çoklu ve alan ilişkisel analiz) ve bu formallik endüstriyel sağlamlılığa giden süreçte oldukça başarılı bir şekilde kullanılmıştır. Konumsal veritabanının standartlaşması (sağlanan konumsal işleçlerin tam ve ortogonal olması) da benzer süreci takip ederek 4-kesişim modelini taban almıştır (Carson, 2000; OGC, 2003). Konumsal sorgu dilinde kullanılan işleçler üç grupta toplanabilir: (a) Topolojik ilişkileri ifade eden konumsal önermeler (b) Kesişim veya birleşme gibi konumsal veri türlerini hem alan (domain) hem değişim (range) olarak alabilen işlevler; başka bir deyişle, konumsal nesneler kümesi üzerinden uygulanabilen konumsal işlevler (c) Sayı döndüren konumsal işleçler. Sorgunun sonucu genellikle konumsal veri nesnelerinden meydana gelen bir küme olmaktadır. Konumsal sorgu türlerine aşağıdaki somut örnekler verilebilir:

(1) Kesin Eşleşleştirme Sorgusu
(2) Nokta Sorgusu
(3) Pencere sorgusu
(4) Kesişme sorgusu
(5) Kapsama sorgusu
(6) Komşuluk sorgusu
(7) En yakın komşu sorgusu
(8) Konumsal birleşim

## 3. KONUMSAL ERİŞİM YÖNTEMLERİ

Literatürde pek çok konumsal erişim yöntemine rastlamak mümkündür. Her bir yöntemden bahsetmek çok yer alacağından bu makalede örnek olarak en çok kullanılmakta olan Izgara Yapısı, Doğrusal Dörtlü Ağaç, R-Ağacı, $R^*$-Ağacı gibi bazı konumsal erişim yöntemlerinden bahsedilecektir.

Konumsal bir dizini [MBB, objectID] çiftlerini yaprak kabul eden bir sıradüzensel yapı olarak görebiliriz. Burada MBB, iki boyutlu geometrik nesnenin en küçük sınırlayıcı dikdörtgeni, objectID ise nesnenin ayırtedici niteliğidir. Eğer iki boyutlu iki nesnenin arasında herhangi bir topolojik ilişki (kapsama, kesişme, örtme vs. gibi) bulunmaktaysa bu ilişkilerin öncelikle MBB' lerde ortaya çıkmaları gerekmektedir. Dizinleme ağacında ara düğümler çocuklarının taradığı alanı kapsamaktadır. Bu kapsam ilişkisi, verilen sorgu penceresi ile kesişen yapraklara (her bir yaprak

konumsal olarak birbirine yakın [MBB, objectID] çiftleri içeren disk sayfasına tekabül eder) bir dal boyunca ilerleyerek erişmeyi sağlar. Dikkat edilmesi gereken husus aday MBB' nin eldeki sorgu penceresi ile kesişmesi onun sınırladığı geometrik şeklin de kesişmesi gerektiği anlamına gelmediğidir. Bunun için ikinci aşamanın (ya da süzmenin) bir parçası olarak, kesişen [MBB, objectID] çiftinin objectID göstergesi kullanılarak onun gösterdiği gemetrik nesnenin şekil bilgisine ulaşılmaya çalışılır ve böylece gerçek kesişmenin olup olmadığı denetlenir.

Konumsal erişim yöntemleri (KEY) ile ilgili iki önemli husus bulunmaktadır (Rigaux, vd., 2002):

- *Zaman karmaşıklığı:* Nokta ve pencere sorgularını yarı lineer zamanda gerçekleştirebilmelidir.
- *Uzay karmaşıklığı:* Boyutu dizinlenen koleksiyonla kıyaslanabilir büyüklükte olmalıdır.

İlk gereksinim anlamı, koleksiyonda n adet nesne olduğundan bunların tamamının gözden geçirilmemesinin sağlanılmasıdır. Uzay karmaşıklığı ise eldeki koleksiyonun altta bulunan fiziksel sistemde ne kadar iyi gösterildiği ile ilgilidir. Yani nesnelerin tekrar tekrar gösterilmesi en az olmalı, idealde her nesne bir dizin girdisi ile saklanılmalıdır. Bu durumda KEY'in zaman karmaşıklığı *logn*, uzaysal karmaşıklığı ise *n* mertebesinde olacaktır.

KEY' lerle ilgili önemli olan bir diğer husus ise devingen olma özelliğidir. KEY'ler yeni nesnelerin veritabanına eklenilmesine ve mevcut nesnelerin veritabanından silinmesine dolayısıyla da dizinlenmiş koleksiyonun büyümesine veya küçülmesine performans kaybına yol açmadan izin verebilmelidirler (Rigaux, vd.,2002).

Geleneksel veritabanlarındaki konumsal olmayan verilerin aksine konumsal verilerde anahtar değerlerinde tam bir sıralamasının olmasını beklemek olası değildir. Bu bakımdan var olan büyük sayıdaki KEY'in takip ettiği iki farklı yol bulunmaktadır (Rigaux, vd.,2002):

- Uzaya dayalı yapılar: Bu tür KEY'ler nesnelerin üzerinde bulunduğu iki boyutlu uzayın dikdörtgensel hücrelere bölümlenmesine dayalıdır. Buradaki bölümleme nesnelerin iki boyutlu uzay üzerindeki dağılımdan bağımsızdır ve nesneler bu hücrelere belirli geometrik kritere göre atanmaktadırlar.
- Veriye dayalı yapılar: Bu tür KEY'ler uzaya dayalı yapıların tersine nesneler kümesinin bölümlenmesine göre organize edilen yapılardır. Burada bölümleme üzerinde bulunulan uzaya göre nesnelerin gösterdiği dağılıma göre olmaktadır.

**a. Uzaya Dayalı Yapılara Örnekler**

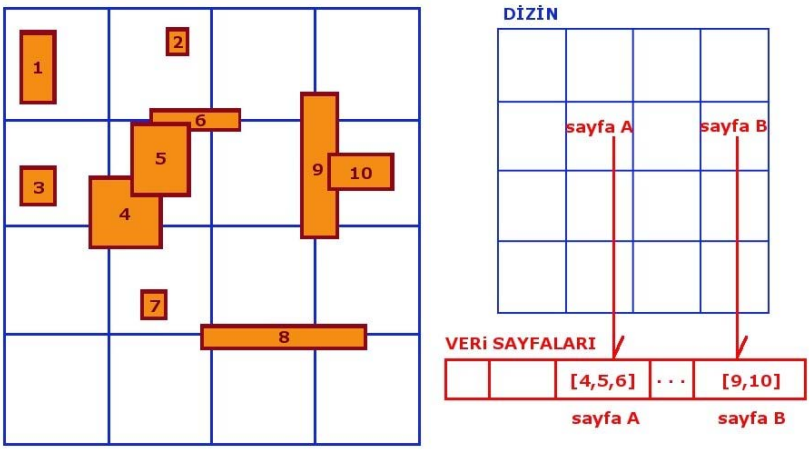
(1) Izgara Yapısı

Arama uzayı ızgara şeklinde dörtgen hücrelere bölümlenmektedir. Sonuçta ortaya çıkan n×m sayıdaki dikdörtgen olmaktadır. Her hücre bir disk sayfası ile ilişkilendirilmektedir. Örneğin *N* noktası *h*. Dörtgen tarafından kapsanıyorsa *N* noktası

*h* hücresine atanmaktadır. *h* hücresindeki nesneler ilgili disk sayfasında bir sıra üzerinde saklanmaktadırlar (Rigaux, vd.,2002).

Bu yapıda Sadece tek bir disk sayfasına ulaşmak yeterli olduğundan nokta sorgusu elverişli olmaktadır. Pencere sorgusunda ise bu pencerenin kaç hücreye yayıldığına bağlıdır ve pencerenin alanıyla doğru orantılıdır (Rigaux, vd.,2002).

Izgara çözünürlüğü dizinlenmesi gerekli olan noktaların sayısına bağlı olmaktadır. *n* adet nokta söz konusu olduğunda *m* nokta kapasiteli sayfalar için en az *n/m* hücreye sahip bir ızgara yapısı gerekmektedir. Ortalamada her hücre m'den daha az nesne bulundurmaktadır. Bu durumda bir hücreye m'den fazla sayıda nokta atandığında bu noktalar taşar. Taşan noktalarla ilişkilendirilen disk sayfası hücrenin asıl sayfasına bağlanılır. Böylelikle bu yapıda dahi tek bir noktaya ulaşabilmek için birden fazla disk sayfasına erişim gerekli olabilir. Eğer noktalar uzayda düzgün olarak dağılmaktaysalar bu durum pek meydana gelmez ve bir noktaya ulaşmak genellikle tek bir hamlede gerçekleştirilebilir. Ancak noktaların belli bölgelerde öbeklendiği bir durumda böyle birşey söz konusu olmamaktadır ve en kötü durumda da tüm noktalar bir hücre içerisinde bulunabilmektedirler. Böyle bir durumda bir noktanın araştırılması doğrusal zaman alacaktır ve bu KEY'lerden istenilmeyen bir özelliktir. Aynı şekilde çok fazla sayıda boş disk sayfası olacağından bellek kullanımı bakımından KEY başarısız olmaktadır (Rigaux, vd.,2002).




Şekil 3. Sabit ızgara yapısı ile dizinleme.

Şekil 3'de sabit ızgara yapısı ile dizinleme görülmektedir. Örnek olarak 10 adet en küçük sınırlayıcı dikdörtgen verilmiştir. Görüldüğü üzere veri dağılımına bağlı olmaksızın uzay 16 hücreye bölünmüştür ve her hücrede bulunan dikdörtgenler o hücreye karşılık gelen sayfalara kaydedilmiştir. Bu duruma örnek olarak şeklin sağ tarafında iki sayfa (sayfa A ve sayfa B) verilmektedir. Bir dikdörtgenin bir hücreye kaydedilmesi için tamamının o hücre içinde bulunuyor olması gerekmemektedir. Örneğin 6 numaralı dikdörtgen aynı anda 4 farklı hücrede yer almaktadır.

Izgara yapısının farklı bir versiyonunda hücre boyutu sabit olmamaktadır. Öyle ki taşmaya sebep olacak şekilde yeni noktalar eklendiğinde hücreler bölünmekte ve

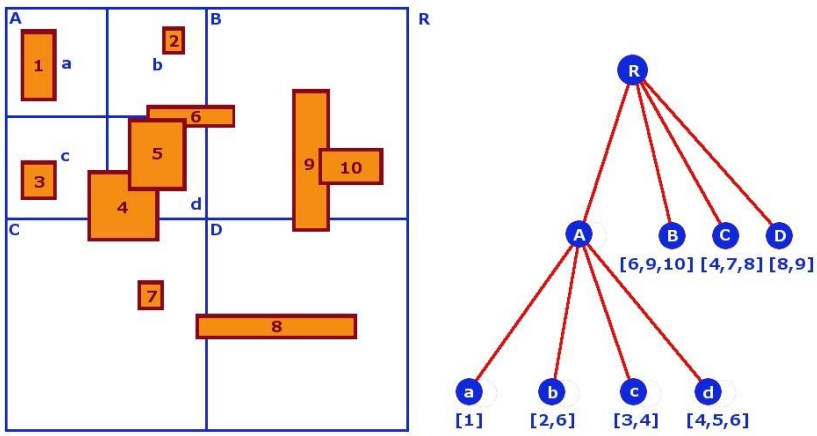
noktalar yeni ve eski hücreler arasında dağıtılmaktadır. Bu şekildeki bir yaklaşım orijinal sabit ızgara yapısını geliştirse de büyük veri kümelerini kapsamak için gerekli hücre sayısı ana belleğe sığamayacak bir büyüklüğe ulaşabilmektedir. Bu olumsuz durum özellikle pencere sorgularında ortaya çıkmakta ve ızgara yapısı için engelleyici bir durum meydana getirmektedir.

(2) Doğrusal Dörtlü Ağaç

Pek çok uygulamada verilerin çekildiği altta yatan uzayı özyineli olarak küçük parçalara bölerek verilerin hiyerarşik olarak temsil edilmesi faydalı olmaktadır ki bu alt bölümlerin oluşturulma kriteri genellikle veri homojenliğine ya da veri dağılımına dayalı olmaktadır (Lee, vd., 2000). Basitliğinden dolayı çokça tercih edilmekte olan doğrusal Dörtlü Ağaç yapısında her bir kare tekrarlı olarak dört alt kareye bölünmekte ve bu işlem bir karedeki nesne sayısı disk sayfasının kapasitesine eşit oluncaya dek sürdürülmektedir. Bu dizin her düğümün dört alt düğüme bölündüğü dörtlü ağaç ile temsil edilmektedir. Ağacın her bir yaprağı ile bir disk sayfası ilişkilendirilmektedir. Bu yapıda her bir MBB kapladığı karelerin tümünde gösterilmektedir. Bu da bir nesnenin dizinde birden fazla kez yer almasına sebep olmaktadır. Nokta sorgularında ağaçtan aşağıya doğru bir yoldan gidilmekte ve işlem logaritmik zamanda gerçekleştirilebilmektedir. Pencere sorguları ise biraz daha karmaşıktır (Rigaux, vd.,2002).

Mevcut bir kareye yeni bir MBB eklendiğinde öncelikle MBB ile kesişen her yaprak ele alınmaktadır. Bu durumda ya boş yer vardır ve MBB buraya eklenir ya da boş yer yoktur (disk sayfası doludur) ve kapasitesi dolu olan yaprak kareler dörde bölünür ve ortaya çıkan yeni ve mevcut kareler arasında yeni ve mevcut nesneler (MBB'ler) yeniden dağıtılır (Rigaux, vd.,2002).

Şekil 4'de dörtlü ağaç ile dizinleme görülmektedir. Örnekte sayfa kapasitesi 4 kayıt olarak alınmaktadır. Bu durumda bir yaprakta en fazla 4 adet dikdörtgen tutulabilmektedir. A karesinde 6 adet en küçük sınırlayıcı dikdörtgen bulunduğuna ve bu sebeple kendi içerisinde dörde bölündüğüne dikkat edilmelidir. Bu yapıda da bir dikdörtgen birden fazla yaprakta tutuluyor olabilmektedir.




Şekil 4. Karesel ağaç ile dizinleme.

Bu dizinleme yapısında veri erişim zamanı ağacın derinliğine bağlı olmaktadır. Durağan durumda ağaç sıkıştırılabilmektedir ancak devingen durumda performans belirgin ölçüde azalabilmektedir. Bunun yanı sıra doğrusal dörtlü ağaçta da ızgara yapısında olduğu gibi dikdörtgenlerin farklı yapraklarda tekrarlama olabilmektedir ve durum veri kümesi büyüdükçe tekrar sayısındaki üssel artıştan dolayı içinden çıkılmaz bir durum alabilmektedir (Rigaux, vd.,2002).

Hafıza gereksinimleri bakımından karesel ağaçta verilen bir derinlik için meydana gelen en kötü durum ele alınan bölgenin dama tahtası yapısına karşılık geldiği durumdur (Samet, 1984).

**b. Veriye Dayalı Yapılara Örnekler**

(1) R-Ağacı

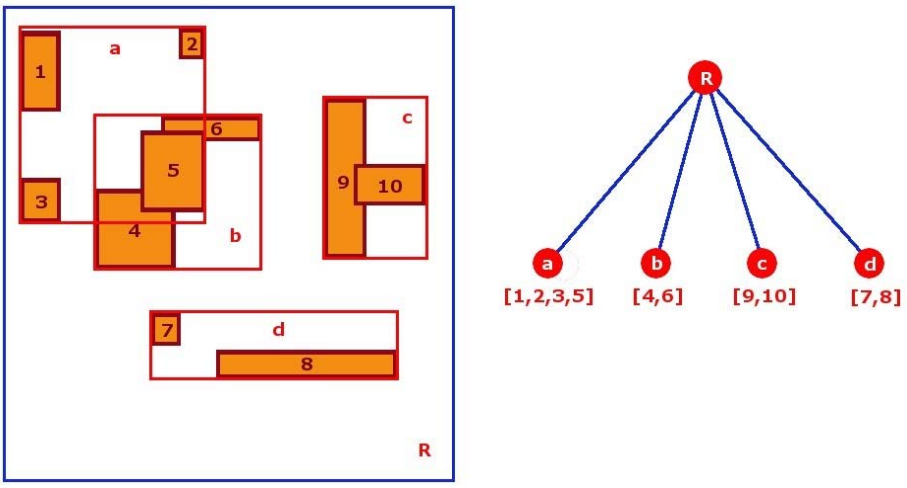
B-Ağacında olduğu gibi alfanümerik bir anahtar yerine R-Ağacındaki her bir düğüm bir dikdörtgen alana sahiptir ve bunun anlamı da sadece bu dikdörtgen tarafından kapsanabilen verilerin düğüme bağlı alt düğümlerde bulunabileceğidir. R-Ağacı, üst düğümlerde bulunan dikdörtgensel bölgelerin bu düğümlere bağlı bulunan alt düğümlerdeki tüm dikdörtgensel bölgeleri tamamiyle kapsayacağı şekilde hiyerarşik bir yapıdadır. Bu özellikler R-Ağacında gerçekleştirilen aramaları basit ve etkili yapmaktadır (Sheng, vd.,1990).

Bir sorgu penceresi içerisindeki tüm verileri bulmak için arama yapıldığında arama algoritması özyineli bir şekilde ağaçta aşağıya doğru ilerler ve bulunduğu düğümdeki dikdörtgensel bölgeyi arama bölgesi ile karşılaştırarak sadece ilgili verilerin bulunabileceği düğümleri araştırır (Sheng, vd.,1990).

R-Ağacı kendisini uzaydaki dikdörtgen dağılımına adapte eden bir yapıdadır. Yaprak olsun ya da olmasın ağaçtaki her düğüm bir disk sayfasına karşılık gelmektedir. Bu ağaç yapısında dikdörtgenler kapsama ilişkilerine göre organize edilmektedirler. Her düğüm bir dikdörtgene karşılık gelmekte, o dikdörtgen içerisinde bulunan diğer kapsayan dikdörtgenler de o düğümün çocuklarını oluşturmaktadır. Her yaprak düğümünde bir dizi [MBB, objectID] çifti bulunmaktadır. Bu her bir ikiliye biz veri girdisi diyeceğiz. Bilindiği üzere MBB yani en küçük sınırlayıcı dikdörtgen $(x_1,y_1,...,x_d,y_d)$ şeklinde bir *d*'lidir (burada *d* uzayın boyutudur).

Dizinlenmiş koleksiyondaki bir dikdörtgene ulaşmak için ağacın kök düğümünden başlanarak alt düğümlerdeki sınırlayıcı dikdörtgenlerle kesişme ya da örtüşme kontrolünün yapılması yeterli olmaktadır. Ağacın derinliği logaritmik olduğundan ve her düğüm bir disk sayfasına karşılık geldiğinden dejenere olmayan durumlarda giriş/çıkış işlemleri koleksiyonun büyüklüğünün logaritması mertebesinde olmaktadır (Rigaux, vd.,2002).

Şekil 5' de R-Ağacı ile dizinleme görülmektedir. Örnek şekilden de görüldüğü üzere her bir en küçük sınırlayıcı dikdörtgen sadece bir adet yaprakta bulunmaktadır. Yine de ağaca yapılan eklemeler sonucunda iki farklı yaprağın örttüğü alanlar çakışabilmektedir (şekilde a ve b yapraklarının çakışması gibi).




Şekil 5. R-Ağacı ile dizinleme

R-Ağacının üst düğümlerindeki iki ya da daha çok en az kapsayan dikdörtgenin üst üste geldiği yer sorgu penceresinin kapladığı alan tarafından örtüldüğünde, arama işlemi ilgili düğümlerin alt ağaçlarına dağılır. Bu da disk erişimlerindeki artışa neden olmaktadır. Çakışma ne kadar artarsa arama sırasında ziyaret edilmesi gereken yolların sayısının artma olasılığı da o kadar çoğalmaktadır (Yu, vd., 1999).

```
R-Ağacı_NoktaSorgulama(N : Nokta) : küme(kayıt_no.)
BAŞLA
        sonuç = Ø

        //Adım 1: Ağacı dolaş ve N noktasını barındıran yaprakların (YK)
        // kümesini döndür

        YK = AğacıDolaş(kök, N)

        //Adım 2: Yaprakları tara ve N'yi barındıranları sakla
        DÖNGÜ YK'deki her bir yaprak (Y) için tekrarla
                //Ele alınan yapraktaki kayıtları (k) tara
                DÖNGÜ Y'deki herbir k için tekrarla
                        Eğer k, N 'yi kapsıyorsa sonuç = sonuç U {k.kayıt_no.}
                DÖNGÜ SONU
        DÖNGÜ SONU

        sonuç'u döndür
SON
```

Şekil 6. R-Ağacında nokta sorgulama algoritması.

Şekil 6'da R-Ağacında nokta sorgulama algoritması görülmektedir. Örnek olarak nokta sorgulama algoritması verilmektedir; pencere sorguları da benzer olarak yapılabilmektedir. Nokta sorgulama algoritması verilen bir noktanın, içerisinde yer aldığı en küçük sınırlayıcı dikdörtgenlerin kümesini döndürmektedir. AğacıDolaş fonksiyonu dolaşmaya başlayacağı düğümü ve aranılan noktayı almaktadır. Burada başlangıç düğümü kök olmaktadır. Bu fonksiyon sorgulaması yapılan N noktasını

barındıran yaprakların kümesini döndürmektedir. Daha sonra her bir yapraktaki kayıtlar (MBB'ler) ele alınmakta ve noktanın içerisine düştüğü kayıtların kayıt numaraları sonuç kümesine eklenmektedir. Bir yaprak tarafından barındırılan bir noktanın o yaprağın tuttuğu dikdörtgenlerin içine düşmesinin zorunlu olmadığına dikkat edilmelidir. Aynı şekilde MBB içerisinde kalan bir nokta zorunlu olarak o MBB'nin içerisinde bulunan geometrik nesne (örneğin bir poligon) ile kesişmemektedir.

Farklı versiyonları da geliştirilmiş bulunan R-Ağacı esas olarak aşağıda verilen özellikleri sağlamaktadır (Rigaux, vd.,2002):

- Ağaçta bulunmakta olan her (kök hariç) düğüm için veri girdi sayısı sayısı m ile M arasındadır. Burada m $\in$ [0, M/2]' dir.
- Yaprak olmayan her düğümdeki veri girdisi diğer kapsayan düğümlerin ya da yaprakların sayfa adreslerini vermektedir.
- Yapraktaki her bir veri girdisinde yer alan (MBB, objectID) ikilisindeki objectID, en küçük sınırlayıcı dikdörtgeni MBB ile verilmekte olan nesnenin adresidir.
- Yaprak olmadığı taktirde kökde en az iki adet veri girdisi bulunmaktadır.
- Bütün yapraklar ağaçta aynı seviyede bulunmaktadır.

Listelenen bu özellikler devingen olarak gerçekleşmekte olan tüm silme ve ekleme işlerimden sonra da korunmaktadır. Uzayı bölümleyen Karesel Ağacın aksine R-Ağacında MBB'lerin yoğunlaştığı kesimlerde birbirlerine yakın pek çok yaprak bulunmaktadır. Karesel ağaçta yüksek yoğunlukta dikdörtgenlerin bulunduğu bölgelerdeki ağacın dalları uzun olabileceği gibi bu bölgelere kıyasla da diğer bazı bölgelere denk gelen dallar kısa olabilmektedir (Rigaux, vd.,2002).

Uzayı bölümleyen konumsal erişim yöntemlerinin aksine bir nesne sadece tek bir ağaç yaprağında bulunmaktadır. Bununla beraber iç düğümlerdeki kapsayan dikdörtgenler uzayın tam bir bölümlemesini oluşturmamaktadırlar ve de üst üste binmiş olabilirler (Rigaux, vd.,2002).

(2) R* Ağacı

R-Ağacının bir versiyonu olan R* Ağacı hem nokta hem de konumsal nesneler için etkili bir yapıdır. Nesneler en küçük sınırlayıcı dikdörtgenler (MBB) ile temsil edilmektedir. Yapraklardaki kayıtlar [MBB, kayıt-göstergeci], yaprak olmayan düğümlerdeki kayıtlar ise [MBB, alt-düğüm göstergeci] şeklinde olmaktadır. Eğer bir düğümde olabilecek maksimum kayıt sayısı M ile gösterilmekte ise o zaman m ≤ M/2 bir düğümde olabilecek minimum kayıt sayısını göstermektedir. m sayısı ağacın dejenere olmasını engellemekte ve etkili depolama uygulamasını sağlamaktadır. Optimizasyon kriteri olarak MBB'lerin çevre uzunluğu değeri kullanılmaktadır. R* Ağacı yükseklik dengeli bir ağaçtır ve zorlamalı tekrar-eklemeyi uygulamaktadır (Venkateswaran, vd., 2006). Zorlamalı tekrar-ekleme, R* Ağacının dinamik olarak yeniden organize edilmesini sağlayabilmek için ekleme işlemi sırasında hali hazırdaki kayıtların bir kısmının yeni kayıt ile birlikte yeniden eklenmesini zorlamaktadır.

Zorlamalı tekrar-ekleme ile ilgili olarak (Beckmann, vd., 1996):

- Zorlanmış ekleme işlemi komşu düğümler arasındaki girişleri değiştirmekte ve örtüşmeyi azaltmaktadır.
- Yan etki olarak depolama daha elverişli hale gelmektedir.
- Yeniden yapılanmanın tesiriyle daha az bölünme meydana gelir.
- Dış dikdörtgenler yeniden eklendiklerinden temel karelerin yapıları daha karesel hale gelir ve bu da arzu edilen bir özelliktir.

R* ağacı, R ağacındaki ekleme algoritmasına çeşitli geliştirmeler sağlamaktadır. Esas olarak, bu geliştirmeler şu üç parametreyi optimize etmek içindir: (1) düğümlerin çakışması, (2) bir düğüm tarafından kapsanan alan ve (3) bir düğümdeki kapsayan dikdörtgenin çevre uzunluğu. Bu sonuncu durum dikdörtgenlerin şeklini temsil etmektedir çünkü belli büyüklükte bir alan verildiğinde dikdörgenin çevre uzunluğu azaltan şekil bir karedir.

Bu her üç parametreyi aynı anda optimize edebilecek iyi oluşturulmuş bir teknik bulunmamaktadır. Dolayısıyla da tasarım yaklaşımı ekleme algoritmasındaki her bir modulü göz önünde bulundurmak ve bunları çeşitli sezgisel yöntemlerle test etmeye dayanmaktadır (Rigaux, vd.,2002).

Düğüm kapasitesini gösteren M değeri aşıldığında yine bir sayfa ayrılmalı ve M+1 sayıdaki kayıt bu iki düğüm arasında dağıtılmalıdır. Çözüm uzayının büyüklüğünün üssel olmasından dolayı tüm olası dağılımlara bakmak mümkün değildir. R* ağacında dikdörtgenlerin bölünmesi gerçekleştirilirken uygulanan yöntemde bir eksen boyunca çizilen doğrudan faydalanılmaktadır. Bu işlem için kullanılacak en iyi eksenin bulunması çok pahalı bir işlem olmamaktadır ve en iyi eksende tüm dikdörtgen çevre uzunlukları toplamı minimal olmaktadır.

## 4. SONUÇ

Bu çalışmada konumsal erişim yöntemlerinden sadece olan Izgara Yapısı, Doğrusal Dörtlü Ağaç, R-Ağacı, $R^*$-Ağacı yöntemlerine değinilmiştir. Zaman içinde geliştirilmiş olan $R^+$-Ağacı, KD2B-Ağacı, Hilbert R-Ağacı, skd-Ağacı, Buddy Ağacı, ve MX-CIF 4'lü ağacı (Kedem, G., 1982) gibi daha bir çok konumsal erişim yöntemleri bulunmaktadır ve geliştirilmeye de devam edilmektedir (Gaede, vd.,1998).

Denektaşı testleriyle konumsal erişim yöntemleri üzerine yapılan performans araştırmaları sonuçlarında herhangi bir konumsal erişim yönteminin diğerlerine karşı belirgin bir şekilde daha iyi olduğu gözlenmemiştir. Buna sebep en iyi performansı tanımlayabilecek çok sayıda kriterin olması ve en iyi performansı hesaplamak için çok sayıda parametrenin var olmasıdır. Bir erişim yöntemi nokta sorguları için çok iyi derecede performans gösterirken, aynı yöntem keyfi dikdörtgen sorguları için oldukça kötü performans gösterebilmektedir.

Konumsal erişim yöntemlerinin seçimlerinde performans kriterinin en az belirleyici özelliklerden biri olduğunu söyleyebiliriz. Konumsal erişim yöntemlerinden hangilerinin ticari kuruluşlar tarafından seçildiğine bakıldığında da, kuruluşların kolay anlaşılan ve kolay elde edilebilen konumsal erişim yöntemlerini seçtiklerini

görmekteyiz. Örneğin, SICAD [Siemens Nixdorf Informationssysteme AG] ‘ın ve Smallworld GIS ‘in dörtlü ağaçları, Informix ‘in R-Ağaçlarını, Oracle ‘ın Z-ordering yöntemlerini seçtiklerini bilmekteyiz (Gaede, vd.,1998).

## K A Y N A K L A R


**Beckmann, N., Kriegel, H. P., Schneider, R., Seeger, B.**, 1996, The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *ACM SIGMOD International Conference on Data Management,* Atlantic City, N.J., Haziran, s. 322-331

**Carson, G.S.** 2000, Spatial Standardization, *ACM SIGGRAPH Computer Graphics*, 34(3): 38-41.

**Clementini, E., Felice, P. D.**, 2000, Spatial Operators. *ACM SIGMOD Record,* **29,** No.3

**Egenhofer, M. J. & Franzosa, R. D.,** 1991, Point-Set Topological spatial relations, *International Journal for Geographical Information Systems*, 5(2): 161-174.

**Gaede, V., Günther, O.**, 1998, Multidimensional Access Methods. *ACM Computing Surveys,* **30,** No.2.

**Güting, R. H.**, 1994, An Introduction to Spatial Database Systems. *VLDB Journal,* **3**, 357-399.

**Guting, R. H. And Schneider, M.**, 1993, M. Realms: A foundation for spatial data types in database systems. *Proceedings of the Third International Symposium on Large Spatial Databases, Singapore*.

**Guting, R. H. And Schneider, M.**, 1995, Realm-Based Spatial Data Types: The ROSE Algebra. VLDB Journal, 4(2):243--286.

**Kedem, G**., 1982, The Quad-CIF tree: a data structure for hierarchical on-line algorithms, Proceedings of the Nineteenth Design Automation Conference, Las Vegas, June 1982, 352-357.

**Lee, M., Samet, H.**, 2000, Navigating Through Triangle Meshes Implemented as Linear Quadtrees. *ACM Transactions on Graphics,* **19,** No.2

**Ooi, B., Tan, K.**, 2001, B Trees: Bearing Trees of All Kinds. *Conferences in Research and Practice in Information Technology,* Melbourne, Avustralya, **5**

**Open Geospatial Consortium (OGC)**, 2003, OGC Reference Model. *OGC 03-040; Version: 0.1.3 (2005) (Çevrimiçi: http://www.opengis.org , Erişim Tarihi: 14.01.2005)*

**Rigaux, P., Scholl, M., Voisard, A.**, 2002, Spatial Databases. Morgan Kaufmann Publishers

**Salzberg, B.**, 1996, Access Methods. *ACM Computing Surveys,* **28,** No.1

**Samet, H.,** 1984,The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys,***16,** No.2.

**Sheng, J., Sheng, O.,** 1990, R-Trees for Large Geographic Information Systems in a Multi-User Environment. *System Sciences*, **2**

**Venkateswaran, J., Subramanya, S. R.**, 2006, Schemes for SR-Tree Packing. *SAC’06,* Dijon, Fransa, 23-27 Haziran

**Yu, B., Orlandic, R., Evens, M.**, 1999, Simple QSF-Trees: An Efficient and Scalable Spatial Access Method. *CIKM’99,* Kansas City, A.B.D.