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

COĞRAFİ BİLGİ SİSTEMLERİNDE GEOMETRİ SINIF KÜTÜPHANESİ

Murat HACIÖMEROĞLU

Yüksek Lisans Tezi
Ankara 2006
BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

COĞRAFİ BİLGİ SİSTEMLERİNDE GEOMETRİ SINIF KÜTÜPHANESİ

Murat HACIÖMEROĞLU

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

Bu tez, 15.06.2006 tarihinde aşağıda üye adları yazılı jüri tarafından kabul edilmiştir.

Ünvan	Adı Soyadı	İmza
Prof. Dr.	Hayri SEVER	.....
Doç. Dr.	Haşmet Gürçay	.....
Yrd. Doç. Dr.	Hasan Oğul	.....

ONAY

/ / 2006

Fen Bilimleri Enstitü Müdürü
Prof. Dr. Emin AKATA

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde beni yönlendiren, yeni yaklaşımlar geliştirme konusunda sürekli fikir ve kaynak sağlayan tez danışmanım, Prof. Dr. Hayri Sever'e, çalışmalarımda desteklerini esirgemeyen Doç. Dr. Haşmet Gürçay ve Güven Köse'ye, bana hep destek olan sevgili babam Selahittin Hacıömeroğlu, annem İnci Hacıömeroğlu, ablam Zeynep H. Gözen, hayat arkadaşım ve eşim Bikem Hacıömeroğlu' na teşekkür ederim.

Ayrıca içinde bu tezin kapsamı da bulunan Evliya Çelebi Coğrafi Bilgi Katmanı TÜBİTAK-105K040 Projesine destek veren TÜBİTAK'a da teşekkür ederim.

ÖZ

COĞRAFİ BİLGİ SİSTEMLERİNDE GEOMETRİ SINIF KÜTÜPHANESİ

Murat HACIÖMEROĞLU

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Ankara, 2006

Coğrafi Bilgi Sistemleri (CBS) dünyada ve ülkemizde sürekli gelişen bir ilgi alanı halindedir. Coğrafi Bilgi Sistemlerinde verilerin yapısı diğer sistemlerden bazı farklılıklar göstermektedir. CBS'ler de veriler konumsaldır (konumsal geometriler) ve geleneksel yöntemlerle işlem yapmak zordur. Birçok araştırmacı geometrik yapılar ve aralarındaki ilişkileri tanımlamak için çalışmalar yapmıştır. Bu çalışmalar nihayet günümüzde ihtiyaçları karşılayabilecek bir standarda kavuşmuştur. Geometrik yapıları ve aralarındaki ilişkileri, geçmiş çalışmalar temelinde standarda kavuşturmak üzere Open Geospatial Consortium (OGC) kurulmuştur. Bu tez kapsamında konumsal verilerin işlenebilmesi için bu verileri tanımlayacak ve aralarındaki işlemleri yürütebilecek konumsal bir sınıf kütüphanesi tam ve doğru olarak gerçekleştirilmiştir.

Söz konusu geometrik sınıf kütüphanesi boyutsal olarak genişletilmiş dokuz kesişim modeli temel alınarak "Microsoft .net" platformunda geliştirilmiştir. İlgili platform web servisleri oluşturmada oldukça esnek bir yapı sunmaktadır. Sonuç sınıf kütüphanesinin sınanması yine *OGC* tarafından standartlaştırılan "iyi tanımlanmış metin" tabanlı *XML* formatında sınama verileri ile gerçekleştirilmiştir.

Anahtar Kelimeler: Konumsal Veri, Geometrik Sınıf Kütüphanesi, Coğrafi Bilgi Sistemi.

ABSTRACT

GEOMETRIC LIBRARY IN GEOGRAPHIC INFORMATION SYSTEMS

Murat HACIÖMEROĞLU

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

Ankara, 2006

Geographical Information Systems (GIS) has been a progressively developing interest area in the world and in our country. In Geographical Information Systems, the structure of the data indicates some differences from other systems. In GIS, data are spatial (spatial geometry) and it is difficult to make operations with traditional methods. Many researchers have made studies to describe the geometric structures and the relationships between them. Today, these studies finally reached the standards that meet the needs. On the basis of previous studies, in order to reach the standards for geometric structures and the relationships between them, Open Geospatial Consortium was founded. In this thesis, a spatial class library was formed completely and accurately in order to describe the data for dealing with this spatial data and to process the operations between them.

The mentioned geometric class library was developed in "Microsoft .net" platform by taking basis from dimensionally extended nine interaction model. The concerned platform offers a very flexible structure to form web services. Testing the final class library was done with "well known text" based; XML formatted testing data which was standardized by OGC.

Key Words: Spatial Data, Geometric Class Library, Geographical Information Systems

İçindekiler

Çizimler Listesi

Tablolar Listesi

Simgeler ve Kısaltmalar

Simge	Açıklama
^{0}	Nesnenin iç kısmı
∂	Nesnenin sınırları
$^-$	Nesnenin tümleyeni
Ø	Nesne kesişimleri boş küme
$\neg$Ø	Nesne kesişimleri boş küme değil
-	Nesne kesişim sonucu önemsiz

Kısaltma	Açıklama
BNF	Bechus-Naur Form
CBS	Coğrafi Bilgi Sistemi
DE-9IM	Boyutsal Olarak Genişletilmiş Dokuz Kesişim Modeli
ESRI	Environmental Systems Research Institute
OGC	Open Geospatial Consortium
SFS	Simple Features Specification
WKT	Well Known Text
XML	Extensible Markup Language

Bölüm 1 - Giriş

1.1 Problemin tanımı

Bilinen anlamda harita kullanımının MÖ 4000'li yıllara kadar uzanmasına karşın, Coğrafi Bilgi Sistemi (CBS) ile ilgili ilk çalışmalar 1960'lı yıllarda ABD'de hükümet ve üniversitelerin birlikte yürüttüğü projelerde, dünya coğrafyasının bilgisayar tabanlı bir sistemde izlenmesi, verilerin derlenip gerektiğinde bunlardan güncel kâğıt haritaların üretilebilmesi yaklaşımı ile başlamıştır. 1970'li yıllarda gelişmesini sürdüren CBS çalışmaları, önce Intergraph arkasından da ESRI (Environmental Systems Research Institute) firmalarının ilk CBS yazılımlarını piyasaya vermesi ile hızlanmıştır. Başlangıçta Intergraph daha çok CBS verilerinin derlenmesi ve bilgisayar kaynaklı haritaların üretimi üzerinde yoğunlaşırken, ESRI ilgili verilerin bilgisayar aracılığı ile değerlendirilip analiz edilmesi konusuna yönelmiştir. Önceleri çok pahalı olan bu ürünler kısıtlı sayıda kullanıcıya erişebilmiştir. 80 ve 90'lı yıllarda Unix iş istasyonlarının ve kişisel bilgisayarların kullanımının artması, daha karmaşık verileri işleyebilen Nesne Yönelimli Veri Tabanlarının ortaya çıkması sonucunda, CBS'ler daha fazla kesim tarafından kullanılabilir hale gelmiştir. Günümüzde ise Internet üzerinden veri akışını kolaylaştıran XML standartları aracılığı ile CBS 'ler artık hemen herkesin kullanabileceği bir teknoloji haline gelmiştir. Halen dünya genelinde CBS yazılımları için her yıl 2-2,5 milyar dolar civarında harcama yapıldığı bilinmektedir.

Konumsal veritabanları günümüzde birçok uygulaması bulunan birbiri ile ilişkili büyük veri derlemlerini sarmalayan coğrafik işlemler kümesidir. Uygulama alanlarından bazıları coğrafi bilgi sistemleri (CBS 'ler), bilgisayar destekli tasarım (BDT), robotlar ve görüntü işleme olarak karşımıza çıkmaktadır. Konumsal veritabanının ilgilendiği *konumsal nesneler* bir saklama biriminde tutulmalı, sorgulanmalı ve görüntülenebilmelidir. Konumsal veritabanları konumsal nesnelerin

ifade şekillerini ve hızlı erişim amaçlı konumsal erişim yöntemleri içeren bununla birlikte hesaplamalı geometri (computational geometry) gibi bazı alanlardan alınmış olan algoritmaların bir araya gelmesi ile oluşan bir teknoloji ürünüdür.

Konumsal veritabanlarının bir uygulama alanı olan Coğrafi Bilgi Sistemleri ise, coğrafi bilgi ile ilişkili olan veriyi alan, tutan, analiz eden ve görüntüleyen bir sistemdir [36] , [22] . CBS'leri Ülkemizde ve Dünyada birçok alanda uygulanmaktadır. Bu alanlardan bazılarını şöyle sıralayabiliriz:

- Şehir ve bölge planlanması,
- Kadastro planlarının yapılması,
- İmar planlarının yapılması,
- Afet bilgi sistemi uygulamaları,
- Askeri Projeler.

Bir CBS 'nin olması gereken özelliklerini tanımlamak için birçok araştırmacı çalışmalar yapmıştır. Esasen bu çalışmalar günümüzde standart haline gelebilecek olgunluğa ulaşmıştır. CBS 'ler üzerinde standartları ortaya koyan OGC (Open Geospatial Consortium) adlı bir uluslararası kuruluş bulunmaktadır. OGC konumsal verilerin işlenmesi, kullanıcıya sunulması ve diğer teknolojik sistemlerle ara yüzü konusunda standartlar belirleyen bir kuruluştur. Bu sebeple uluslararası arenada kendine pay bulmak isteyen tüm CBS geliştiricileri bu OGC standartlarına uymak zorundadır. Bu söz konusu standartlar hakkında gerekli bilgi "*OGC Reference Model"* adlı yayından elde edilebilir [33] . Ülkemizde kendi CBS yazılımını geliştiren kuruluşlara baktığımızda bu standartlara uyan bir CBS 'in var olduğunu görememekteyiz.

Bu tez kapsamında bir CBS 'nin veri alan, bilgiyi görüntüleyen kısımlarından ziyade, CBS 'lerin konumsal veritabanında olması gereken veri türleri ve bu veriler arasındaki operasyonlar ile ilgilenilmektedir. Esasen bir Coğrafi Bilgi Sistemi temel olarak konumsal verileri depolama ve işleme yeteneğine ihtiyaç duyar. Konumsal ve konumsal olmayan verilerin birlikte bir veritabanında tutulması ve birlikte

işlenebilmesi gerekmektedir. Konumsal ve konumsal olmayan verilere örnek vermek gerekirse;

"İçme suyu kaynaklarının 10 Km. yakınındaki yerleşim bölgelerini bul."

Yukarıdaki sorguyu ele alacak olursak içme suyu kaynaklarını uzaydaki koordinatları, bu kaynaklara 10 km. yakınlıkta olan yerleşim bölgelerinin koordinatları, yine bu yerleşim bölgelerinin alanları gibi veriler konumsal verilere girmektedir. İçme suyu kaynaklarının çeşitleri, isimleri, yine bu içme suyu kaynaklarına 10 km. yakınlıkta bulunan yerleşim alanlarının türleri (köy, ilçe, şehir…), nüfusları, isimleri gibi bilgiler ise konumsal olmayan veri türlerine girmektedir.

Görüldüğü gibi konumsal ve konumsal olmayan veriler birbirlerinden ayrılamaz veri türleridir. Eğer yukarıda örnek olarak verilmiş sorgunun sonucu olarak sadece konumsal veriler döndürülseydi, anlamsız koordinat verileri kümesi ile karşılaşmış olurduk.

Konumsal veritabanlarının temel sorunlarından biri de konumsal ve konumsal olmayan verilerin birlikte tutulup tutulmayacağıdır. Eğer konumsal ve konumsal olmayan veriler birlikte tutulacaksa bu geleneksel veritabanlarını kullanamayacağımız ve baştan sadece konumsal verileri işlemek amacıyla oluşturulması gereken veritabanı yönetim sistemleri geliştirilmesi gerektiği anlamına gelir. Eğer konumsal ve konumsal olmayan veriler ayrı olarak tutulacaksa bu kez de bu verilerin işlenirken birleştirilmesi sorunu ortaya çıkmaktadır. Coğrafi bilgi sistemlerinin ortaya atılmasından bu yana tartışılan bu ikilem aslında günümüzde basit bir tercih sorunu haline gelmiştir. Konumsal ve konumsal ve konumsal olmayan verileri birlikte tutmak tercihine **gömülü sistem**, yine konumsal ve konumsal olmayan verilerin ayrı olarak tutulmasına da **ayrık katmanlar sistemi** olarak adlandırılacak.

1.1.1 Gömülü Sistemler

Günümüzde bu sistemlere verilebilecek başarılı örnekler bulunmaktadır. Bunlar **Oracle Spatial, DB2 Spatial** olarak örneklenebilir**.** Gömülü Sistemlerde konumsal olmayan ve konumsal veriler aynı tabloda tutulmaktadır. Bunu başarabilmek için de konumsal veriler için ayrı veri tipleri geliştirilmekte ve veri tabanı yönetim sistemi de bu türleri işleyebilir haldedir. Yukarıda ki örneği tekrar ele alacak olursak;

"İçme suyu kaynaklarının 10 Km. yakınındaki yerleşim bölgelerini bul."

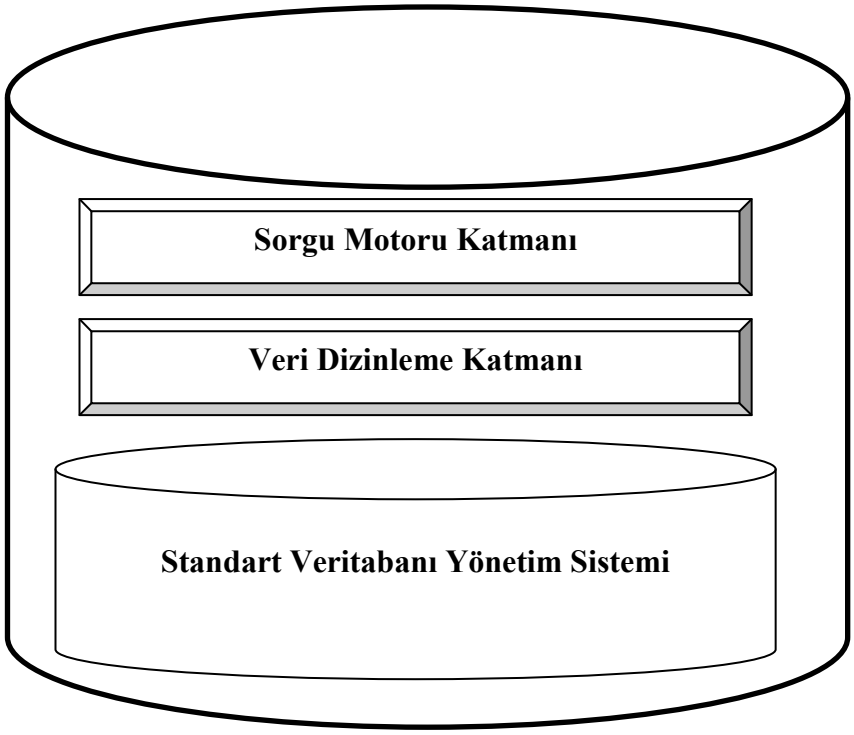
Yerleşim bölgelerinin ve içme suyu kaynaklarının konumsal bilgileri ile konumsal olmayan bilgileri (isim, tür, nüfus…) aynı veri kümesinde tutulmaktadır. Veriler işlenirken veritabanı yönetim sistemin sorgu işleyicisi sadece uzaklık hesaplayıp konumları getirmekle kalmamakta aynı zamanda aynı veri kümesinde bulunan konumsal olmayan verileri getirebilmektedir.

Kısacası gömülü sistemler yeni konumsal sorgulama dilini destekleyen, buna bağlı olarak veri erişim mekanizmalarını ve dizinleme yapılarını konumsal veriler için yapılandırılmış ve grafik işleme çekirdekleri olan özel veritabanı yönetim sistemleridir.

1.1.2 Ayrık Katmanlar Sistemi:

Günümüzde bu sistemlere verilebilecek başarılı örnekler bulunmaktadır. Bunlar ArcGIS (ESRI), PostGis, MySQL Spatial Extender olarak örneklenebilir. Ayrık Katmanlar Sisteminde Konumsal ve konumsal olmayan veriler ayrık veri kümelerinde tutulmaktadırlar. Esasen kullanılan veritabanı yönetim sistemi var olan standart Ansi SQL veri tabanı yönetim sistemlerinden herhangi biridir. Bu veritabanı yönetim sistemleri konumsal veriler için tasarlanmamıştır. Bu nedenle konumsal verilerin ayrı bir yapıda tutulması gerekmektedir. Standart veri tabanı yönetim sistemlerinde konumsal veriyi dizinleyecek ve işleyecek bir yapı bulunmamaktadır. Bir CBS' de konumsal verileri dizinleyecek ve işleyecek becerilerin bulunması

gerekmektedir. Bu durumda Ayrık Katmanlar Sistemi tercih edilecekse dizinleme ve veri işleme katmanları standart veritabanı yönetim sisteminin üzerine geliştirilmelidir.



Şekil (1.1) 1 Ayrık Katmanlar Sistemi Çekirdeği

Şekil (1.1) 1'de görüldüğü gibi Ayrık Katmanlar Sistemi tercih edildiğinde iki kritik katman geliştirilmelidir. Bu tez kapsamında **Sorgu Motoru Katmanı** ile ilgilenilmektedir.

1.1.2.1 Sorgu Motoru Katmanı:

Konumsal verileri işleyebilme yeteneği geleneksel VTYS'lerde bulunmamaktadır. Bir coğrafi bilgi sistemi geliştirmek istendiğinde firmaların bir kısmı kendi veritabanı yönetim sistemlerini geliştirmek yolunu seçmektedirler. Bu zahmetli ve çoğunlukla da geleneksel VTYS'ler kadar kararlı ve güvenilir olmamaktadır. Bunun nedenleri açıktır; geleneksel veritabanı yönetim sistemleri (Oracle, MS SQL) bilgi teknolojileri ile birlikte paralel gelişmiş ve yılardan beri

belirli bir kararlılığa ulaşmış sistemlerdir. Gerek güvenilirlik, gerek hız ve gerekse kararlılık yönlerinden geleneksel sistemlere yaklaşabilecek veri yönetim sistemleri geliştirmek zordur. Hele bir de esas amaçlanan veritabanı yönetim sistemi yapmak değil de coğrafi bilgi sistemi yapmak olunca, vakit ve zaman kaybı ciddi düzeylere ulaşmaktadır.

Bu sebeplerden dolayı var olan geleneksel bir veritabanı yönetim sisteminin üzerinde çalışacak bir ara katman geliştirilme yolu birçok CBS için daha uygun bulunmaktadır.

Yakın zamanlarda dikkatler konumsal veritabanlarına yoğunlaşmıştır ki bu veritabanları geleneksel ve konumsal verileri birleştirebilmektedirler. Bununla birlikte coğrafi bilgi sistemleri gibi veritabanlarında konumsal sorgulama dilinin gerekliliği ortaya çıkmıştır [20] . Konumsal verileri işlemek için standart sorgulama dillerini (örneğin SQL) kullanamayız. Bunun nedeni standart sorgu dillerin konumsal verileri işleyebilmek için gerekli özellikleri taşımamasıdır [11] .

Konumsal verinin geometri ve grafiksel gösterim gibi ek bazı özellikleri bulunmaktadır ki bu özellikleri kullanıcı bir sorgu dili kullanırken adreslemelidir [11] . Konumsal ilişki ve işleçlerin önemi burada ortaya çıkmaktadır ve ilişkisel cebir konumsal özelliklerle genişletilip ortaya Konumsal-İlişkisel Cebir (Geo-Relational Algebra) [28] çıkmıştır.

Yeni bir sorgulama dili geliştirmek yerine var olan veritabanı sorgu dili konumsal olgularla genişletilebilir. Taban sorgulama dili olarak SQL seçilmiş ve ortaya çıkan dile de Konumsal SQL adı verilmiştir. SQL dilini genişletip özel alanlarda kullanma girişimleri diğer çalışma alanlarında bulunmaktadır (Ör. Tarihsel Sorgulama Dili HSQL [37]).

SQL i genişletmeye yönelik Max J. Egenhofer bir model önermiştir [11] . Bununla birlikte coğrafi uygulamalar için klasik VTYS'yi genişletmeye dair OOi, Sack ve McDonell bir klavuz geliştirmiştir [34] . Bu çalışma ayrık katmanlar sisteminin nasıl gerçekleşebileceğini açıklamaktadır.

Konumsal SQL dilinin sahip olması gereken özellikleri OGC (Open Geospatial Consortium) tarafından Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1 [35] adlı yayımlanan belgede açıkça belirtilmektedir. Yani esasen konumsal bir sorgu dilini işlevsel hale getirebilmek için iki temel aşamadan geçilmelidir. Birincisi sorguyu ayıklayıp, konumsal ve konumsal olmayan sorguları birbirinden ayırıp sonuçları ilişkisel cebir ile birleştirmek, ikincisi ise ayıklanan konumsal sorguları işlemek üzere bir sınıf kütüphanesi oluşturmaktır. Bu sınıf kütüphanesinin temel yapısı yine Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1 [35] adlı belgede verilmiştir. Bu sınıf kütüphanesinin coğrafi verilerin işlenmesinde büyük önemi vardır ki bu önem onun tam ve gerçekçi (veya doğru) olmasından kaynaklanmaktadır.

Elbette ki bu Geometrik Sınıf Kütüphanesi tek başına bir şey ifade etmeyebilir. Daha öncede belirtildiği gibi dış dünyadan gelen konumsal sorguları konumsal ve konumsal olmayan sorgular olarak birbirlerinden ayrılmalı daha sonra da bu kütüphane aracılığıyla konumsal sorgular işlenmeli ve son olarak da sonuç kümeleri birleştirilmelidir.

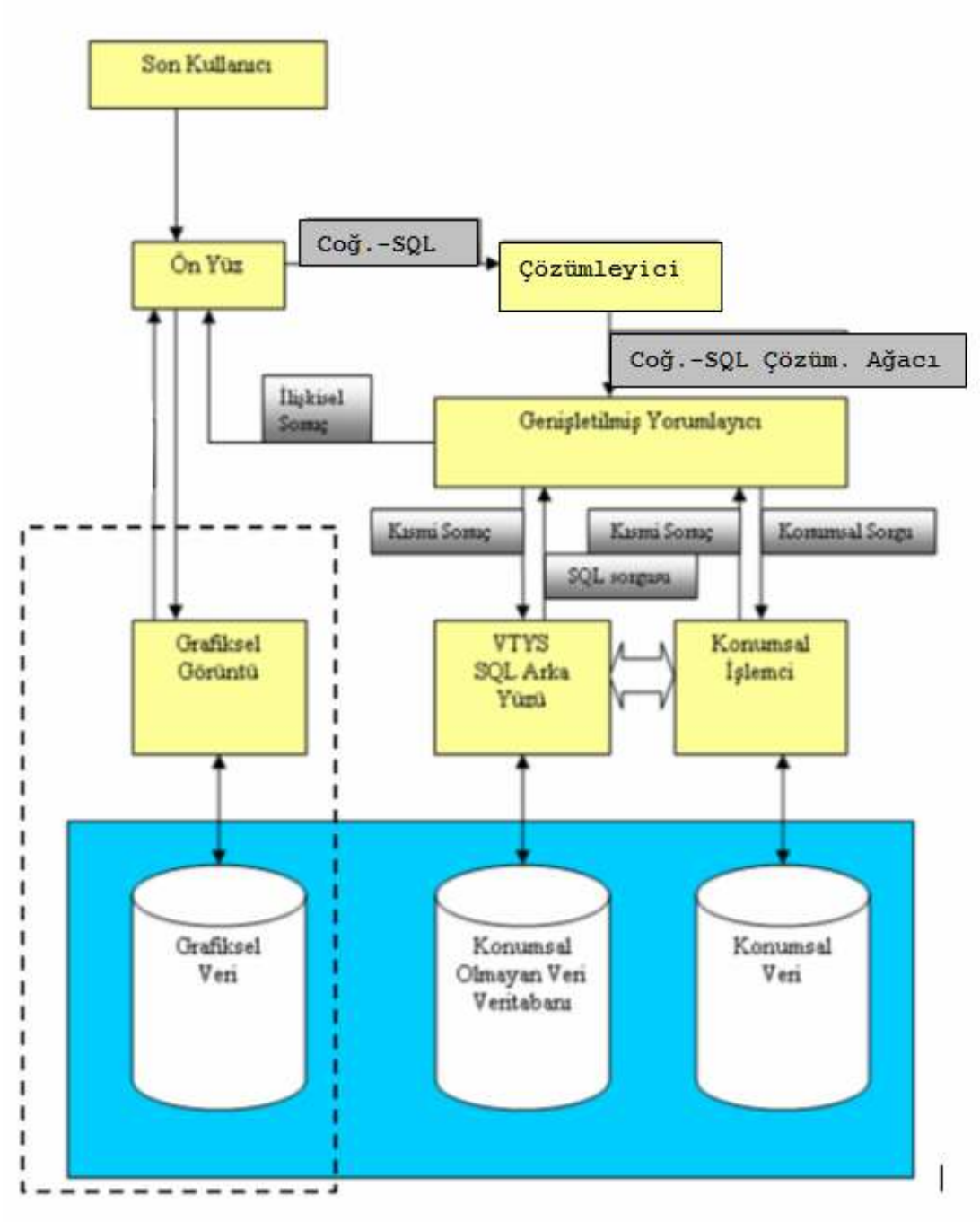
Bir Konumsal-SQL örneği verecek olursak:

```
SELECT demiryolu, isim
FROM   yol,demiryolu
WHERE  yol.isim = 'E5' and yol.intersects(demiryolu).
```

Bunun gibi sorgular CBS'de oldukça yaygındır ve bu gibi sorgulara verilecek cevap konumsal VT katmanının bir sorgu işleme yolu bulmasına bağlıdır. Bu soru daha çok en iyileştirme ile bağlantılıdır. Seçimin ve izdüşümün nasıl yapılacağı varsayılmadan bu sorguyu işlemek için birçok dünya çapında strateji bulunmaktadır. FROM cümleciğinde verilen kartezyen çarpımını aldığımızı varsayalım ve bunun üzerinden 'E5' yol isimli tüm varlıkları seçelim; kesen demiryollarını bulmak için iki teknik bulunmaktadır. İşlenecek demiryolları sayısını bulmak için bir konumsal dizin kullanılabilir, ya da her demiryolu sınanmalıdır. Eğer yol ilişkisi çok büyük ve isim özelliğinde hiç dizin bulunmuyorsa, önce konumsal dizinlemeyi kullanarak kesişme

testini yapmak daha etkin bir yol olacaktır. Bir Konumsal VT Katmanı, bu seçeneklerden en iyisini seçebilme yeteneğine sahip olmalıdır.

Daha öncede belirtildiği gibi, geleneksel VTYS'ler konumsal yapıları ve işlemleri desteklemez. Bu durumda ana görevi, konumsal önermeleri (konumsal) dizinleri kullanarak işlemek olan bir konumsal işlemci gerekmektedir. Bu alt sistem SQL arka son ile ara yüz oluşturacak ve gerekli konumsal işlemleri gerçekleştirecektir. Genişletilmiş sistem mimarisi Şekil (1.2) 2'de verilmiştir.



Şekil (1.2) 2 Genişletilmiş sistem mimarisi.

SQL Arka Yüzü CBS'lerin önemli bir kısmıdır. Burada Coğrafi-SQL sorgusu alt sorgulara bölünür. Bu alt sorgular tamamen konumsal ya da konumsal olmayan sorgular olabilir. Tamamen konumsal olmayan sorgular normal SQL arka son işleyicisinde işlem görebilir. Konumsal sorgular ise konumsal işlemcide işlem görür. Alt sorgular oluşturulduktan sonra, belirli bir sırada işleme sokulur (Alt Sorgu Planı).

Bu sorgu planındaki temel amaç genel işleme maliyeti azaltmaktır. Strateji dört ana adımdan oluşmaktadır.

1.Mantıksal Dönüştürüm.

2.Ayrıştırma.

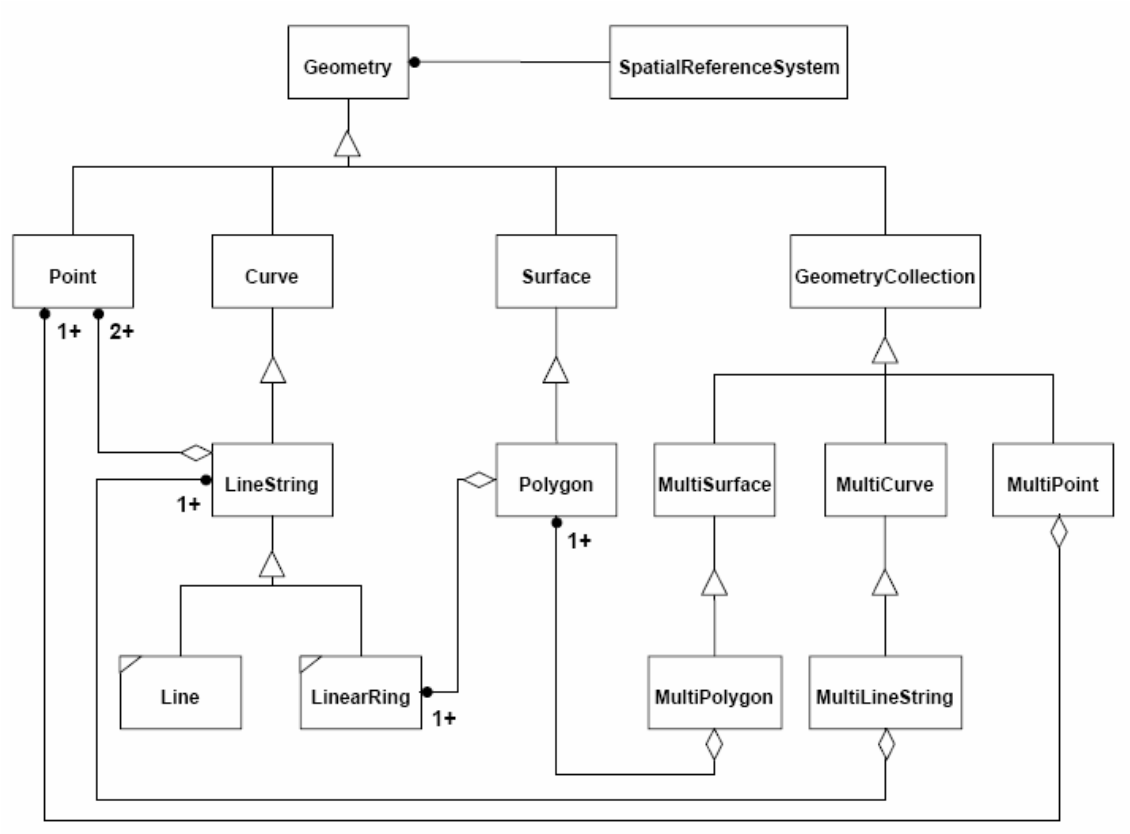
3.Alt Sorgu Sıralama Formülasyonu.

4.Seçim.

Bu aşamalar detaylı olarak Ek—4 Sorgu Motoru Aşamala bölümünde anlatılmaktadır.

Konumsal sorgu motoru konumsal sorguları ayrıştırdıktan sonra onları işleyebilmek için konumsal veri türlerini tanımlayan ve işleyebilen bir kütüphaneye ihtiyaç duymaktadır. Bu tez kapsamında *Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1'de* [35] anlatılan, bir coğrafi bilgi sisteminin en temel ve önemli kısmı olan standart konumsal veri kütüphanesi gerçekleştirilecektir.

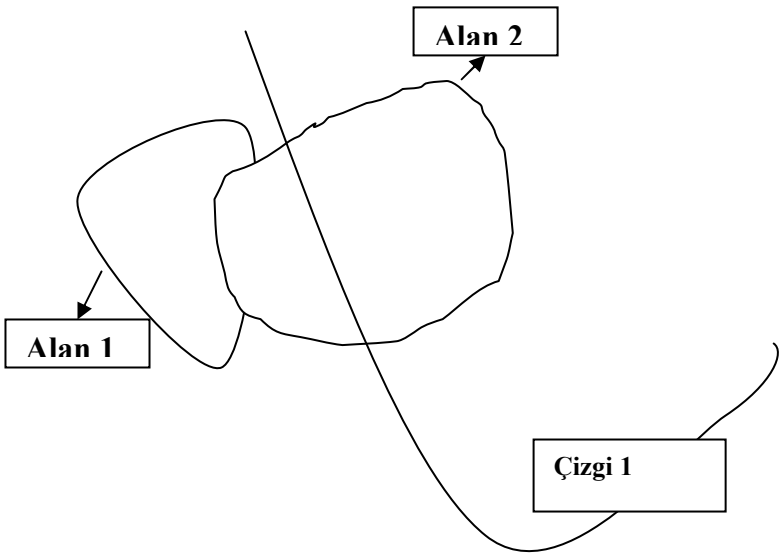
1.2 Tezin Kapsamı Geometrik Sınıf Kütüphanesi



Şekil (1.3) 3 ***Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1 [35] Geometrik Sınıf Kütüphanesi***

Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1'de [35] konumsal sorgu standartları ortaya konulduğu gibi nesne modeli de açıkça anlatılmaktadır (Şekil (1.3) 3). OpenGIS bu nesne kütüphanesini tamamen nesneye dayalı tasarlamıştır. Bunun nedenleri, nesneye yönelik programlama dilleri ve ortamlarının popüler olmasının yanında tüm geometrik veri yapılarının aslında bir *geometry* olmasını yani birbirlerine dönüşebilir olabilmelerini sağlamaktır. Gerçekten de bu gerek diğer kaynaklardan gelen verilerin türünün önceden bilinememesi, gerekse veri tabanındaki verilerin tür dönüşümleri açısından bu çok faydalı olabilmektedir.

Esasen OGC konumsal veri yapısını açıkça göstermiştir. Bu sınıf yapısında her nesne kendi türünün özeliklerini tutmakla sorumlu olduğu gibi, kendi türünden ya da farklı türlerden nesneler ile etkileşimlerinden de sorumludur. Bu etkileşimlere örnek vermek gerekirse:



Şekil (1.4) 4 Değişik Geometrik Şekiller ve Etkileşimleri

Şekil (1.4) 4'de 3 adet geometrik şekil görünmektedir. Bu geometrik şekillerden Çizgi 1 ve Alan 1 Alan 2 ile etkileşmektedir. Bu etkileşim tipinin ve koordinatlarının tespit edilmesi gerektiğinde Geometrik Sınıf Kütüphanesine başvurulacaktır.

Geometrik Sınıf Kütüphanesi OpenGIS tarafından standartlaştırılmadan önce birçok araştırmacı tarafından çeşitli araştırmalar yapılmıştır ve bu çalışmaların sonucunda bu sınıf kütüphanesi çatısı ortaya çıkmıştır.

Öncelikle bu sınıf kütüphanesinin amacının geometrik verileri sorgulama ve işleme olduğu anlaşılmalıdır. Konumsal verilerin sorgulanabilmesi için öncelikle geleneksel verilen sorgulama dili olan SQL taban alınmış ve SQL'e çeşitli eklentiler tanımlanmıştır [11] . Bilindiği gibi SQL dilinin matematiksel ifadesi olan ilişkisel

cebir (ya da ilişkisel çoklu analizi modeli) bulunmaktadır. Öyleyse konumsal sorgulama dilinin de bir konumsal cebire ihtiyacı vardır. İşte bu noktada Guting [24] tarafından geliştirilen ROSE Cebiri ortaya çıkmıştır. ROSE Cebiri bilgisayarlı hesaplamayı kolaylaştırmak için geliştirilmiş Realm [27] veri yapılarını taban almıştır. Realm'ler tüm veri yapılarını bir ızgaraya oturtan ve veri değerlerinin bu ızgara noktası dışında bir nokta olmasını engelleyen koordinat ve veri türleri tanımıdır. Yine ROSE Cebiri gösterim olarak kendine Second-Order Signiture [30] gösterim modelini seçmiştir. Esasen Rose Cebiri konumsal bir sorgulama dilinin matematiksel modelidir.

Veri yapılarının aralarındaki ilişkilerinin tıpkı klasik matematikteki toplama çıkarma işlemleri gibi formalize edilmesi gerekliliği de elbette ki gözden kaçırılmamalıdır. Öncelikle 4-kesişim modeli [18] geliştirilmiştir. Bu model nokta kümesi topolojisini [18] taban almakta ve geometrik nesnelerin içlerini ve sınırlarını karşılaştırarak sınıflandırma yapmaktaydı. Daha sonra nesnelerin tamlayanlarının (iç ve sınırının dışında) da bu sınıflandırmada gerektiği ortaya çıktı [12] ve Egenhofer Dokuz Kesişim Modeli geliştirdi [12] . Dokuz kesişim modeli konumsal veri yapılarının aralarındaki geçerli ilişkileri tanımlamaktadır. Daha sonra Clementini [4] tarafından kesişimlerin dönen sonuçlarını boyutlarına göre de sınıflandıran Boyutsal Olarak Genişletilmiş Dokuz kesişim Modeli geliştirildi [4] (DE-9IM). Günümüzde en gelişmiş ilişkisel kesişme modeli DE-9IM'dir ve konumsal sınıf kütüphanesi bu kesişim modelini temel almaktadır.

Bu noktadan sonra karşımıza konumsal verilerin geleneksel veritabanlarında nasıl tutulacağı sorunu çıkmaktadır. Bir geometrik cisim için ne kadar veri tutulacağı önceden kesin olacak bilinememektedir. Öyleyse Şekil (1.5) 5 deki gibi klasik tablo oluşturup verileri onun içine atma yaklaşımı konumsal veriler için uygun değildir.

Şekil (1.5) 5 Geleneksel bir veri tabanı tablosu örneği

Bu soruna yine OpenGIS bir çok CBS geliştiricisi ile birlikte geometrik verileri tek bir tablo alanında toplama çözümünü getirmiştir. Konumsal veri türleri örneklerinin sayısı önceden bilinmeyen bir sistemde bu verilerin tek bir tablo alanında tutulmasını sağlayan standardın adı ise [33] Well-Known Text'dir (WKT). WKT geometrik verilerin Şekil (1.6) 6'deki gibi (örneğin, Ankara İl Sınırı, Tuz Gölü'nün Topolojik Şekli, apartmanların koordinatları, haritada bir nokta, vb.) tek bir yalın katar şeklinde tutulmasını sağlamaktadır.

GID	*Xmin*	*Ymin*	*Xmax*	*Ymax*	*WKT_Geometry*
1	5	5	3	6	LINESTRING(0 0, 10 10, 20 25, 50 60)
1	-	-	-	-	POINT(15 20)

Şekil (1.6) 6 WKT kullanan konumsal bir veri tabanı örneği

Yukarıdaki nedenlerden dolayı WKT'lerden Geometrik nesneler oluşturalacak bir yordamlar kümesinin Geometrik Sınıf Kütüphanesi ile uyumlu bir sürümünün yazılması gerekliliği ortaya çıkmıştır (Evliya-WKT Sınıfı).

Geometrik sınıf kütüphanesinin ihtiyaç duyduğu birçok geometrik algoritma bulunmaktadır (örneğin; Dış bükey kabuk bulma Graham Scan algoritması [23] Nokta Poligonun İçinde mi? [9] …) Bu algoritmaların hayata geçirilmesi ve sınanması yine bu tez kapsamında bulunmaktadır.

Tüm bu çalışmaların sonucunda geometrik sınıf kütüphanesi standartları ortaya çıkmıştır. Bu standartları ortaya çıkaran ve yukarıda değinilen çalışmaların anlaşılması ve özümsenmesi, sınıf kütüphanesinin geliştiricisi için vazgeçilmez bir ön koşuldur. İleride Geometrik Sınıf Kütüphanesi Standartlarının değişmeyeceği garanti edilemez, araştırmacılar daha iyi modeller ortaya atabilirler, ama elbette ki bu modeller eski modelleri taban alacak modeller ve teoremler olacaktır. Bu durumda hangi modeller, veri yapıları ve nesne uzayları üzerine geliştirilme yapıldığının iyi anlaşılması sadece geometrik sınıf kütüphanesi geliştiricileri için değil, tüm CBS geliştiricileri için bir şarttır.

1.3 Tez Çalışma planı

Bu Yüksek Lisans Tezine Evliya Çelebi Coğrafi Bilgi Katmanı TÜBİTAK-105K040 Projesinin Önerisi Hazırlanarak başlanmıştır. Evliya Çelebi Coğrafi Bilgi Katmanı TÜBİTAK-105K040 Projesi TÜBİTAK BİLİMSEL VE TEKNOLOJİK ARAŞTIRMA PROJELERİNİ DESTEKLEME PROGRAMI'NDAN kabul görmüş ve bu tez yazılırken halen devam eden bir projedir. Proje önerisinde bir coğrafi bilgi sisteminin çekirdeğinde bulunması gereken parçaların nasıl yazılması gerektiği araştırılmış ve gerçekleştirme aşamaları planlanmıştır. Araştırmalar sonucunda geliştirme sürecinin iki ana başlık altında toplanmasının uygun olacağı kararlaştırılmıştır bunlar; sorgu motoru ve dizinleme motorudur. Sorgu motoru katmanında ise, yine iki ana başlık olması gerekliliği ortaya çıkmıştır bunlar;

konumsal sorgu yorumlayıcı ve geometrik sınıf kütüphanesidir. Esasen geometrik sınıf kütüphanesinin günümüzde belirli bir standarda oturmuş olduğunu görülmesine rağmen [33] bu standartların nasıl ve neden ortaya çıktığı araştırılmıştır.

Bu çalışmaların ardından Geometrik Sınıf Kütüphanesinin gerçekleştirme sürecine girilmiştir. Bu süreçte bazı temel geometrik algoritmaların kıyaslanması ve en uygununun seçilmesi gerekliliği ortaya çıkmıştır.

Son olarak da Geometrik Sınıf Kütüphanesinin doğruluğu test edilmiştir. Test verileri VIVID Solutions Inc. adlı kurumdan XML formatında edinilmiş ve uygulamaya konmuştur.

1.4 Tez Düzeni

Bu tez dokümanı içerisinde öncelikle tezin konusu ve kapsamı anlatılmaktadır. Tezin konusu kısmında problemin genel tanımı anlatılmakta, tez kapsamında ise literatür taramasında ortaya çıkan önemli olgulardan ve gerçekleştirilen aşamalardan bahsedilmektedir.

İkinci kısım da ise tez kapsamında geliştirilen Geometrik Sınıf Kütüphanesi Standartlarının nasıl ortaya çıktığı, hangi modeller ve kavramlara dayandığı anlatılmaktadır.

Üçüncü kısımda Geometrik Sınıf Kütüphanesinin detaylı olarak içyapısı ve içerdiği metotlar anlatılmaktadır.

Dördüncü kısım olan Algoritmalar kısmında ise Geometrik Sınıf Kütüphanesi gerçekleştirilirken kullanılan önemli algoritmalar anlatılmaktadır.

Beşinci kısımda, gerçekleştirilen sınıf kütüphanesinin birim testlerinin nasıl yapıldığından ve test verilerinin içyapısından bahsedilmektedir.

Son olarak da atıfta bulunulan çalışmaların tam referansları ve ekler bulunmaktadır.

Bölüm 2 - Konumsal Geometriler

Konusal sistemlerin, ana girdileri olan geometrik verileri tanımaları ve işleyebilmeleri için geometrik veri yapılarının ve aralarındaki işlemlerin tanıtılması gerekir. Birçok araştırmacı hesaplamalı geometrik işlemler için bu veri yapılarını tanımlamış ve aralarındaki işlemleri formalize etmek için çeşitli modeller ortaya koymuşlardır.

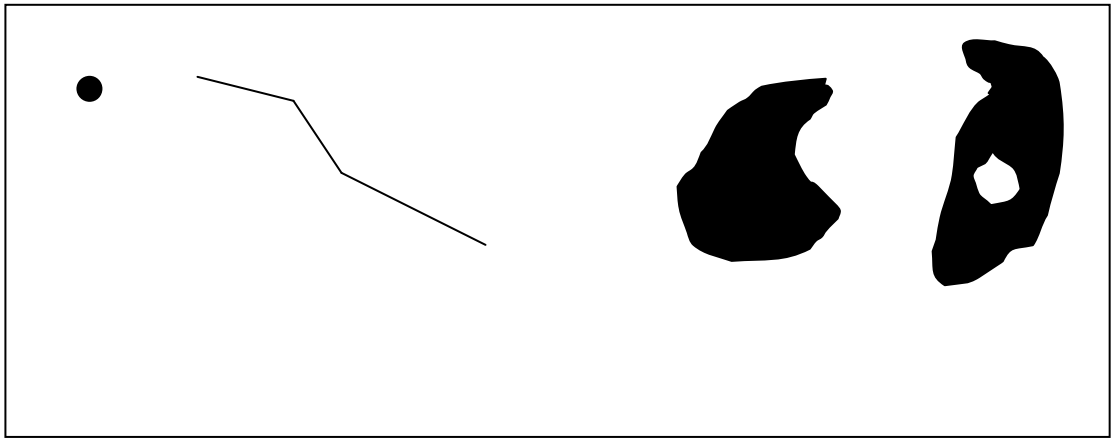
Birçok alanda geometrik, coğrafi ve konumsal veriyi işleme ihtiyacı bulunmaktadır. Konumsal Veri Tabanı terimi son yıllarda oldukça popüler hale gelmiş, bu konudaki çalışmalar her geçen gün artmaktadır. Bu terim esasen uzaydaki iyi tanımlanmış nesnelerin bir arada bulunduğu veritabanlarını ifade etmektedir.

Ralf Hartmut Güting'e [25] göre konumsal veritabanının tanımı şöyledir:

1. Konumsal veritabanı sistemi geleneksel bir veritabanı sisteminin özelliklerine sahip olmalıdır.
2. Veri modelinde ve sorgulama dilinde konumsal veri türlerini kullanır.
3. Konumsal veri türlerini işleyebilen tam ve doğru işlem kümesine sahip olabilmeli ve konumsal dizinleme yapabilmelidir.

Şimdi bu maddeleri ele alacak olursak, ilki basit bir madde gibi görünse de aslında konumsal veritabanı sisteminin esasen sadece özel bir veritabanı olmadığını standart veri modellemesi ve sorgulama işlemlerini yapabildiğini anlatmaktadır. İkinci madde de ise konumsal veri tiplerinin (ör, nokta, çizgi, alan) sistem tarafından tanımlanmış olması ve buna ek olarak da konumsal ilişkileri tanımlayabilmesi anlatılmaktadır. Üçüncü madde ise, büyük nesne veritabanlarının içinde bir kısmını tüm alanı taramadan nesnelerin uzaysal yakınlığı gözetilerek getirilebilmesi gerektiği anlatılmaktadır. Bunun için elbette iyi bir konumsal dizinleme tekniğine ihtiyaç vardır.

Konumsal veritabanlarını modellemek için ayrık nesnelerin tanımlanması ve her birinin geometrik yapısının bilinmesi gerekmektedir. Tek bir nesneyi tanımlamak için esas bileşenler Şekil (2.1) 7'da örneklenmiş olan nokta, çizgi ve alandır. Bir nokta sadece kendi yerini gösteren bir nesnedir. Bir çizgi bir yerden başka bir yere hareketi ve ya bağlantıyı gösteren bir nesnedir. Alan ise, 2 boyutlu uzayda kapsamı olan bir nesnedir.



Şekil (2.1) 7 temel geometrik nesne nokta, çizgi ve alan.

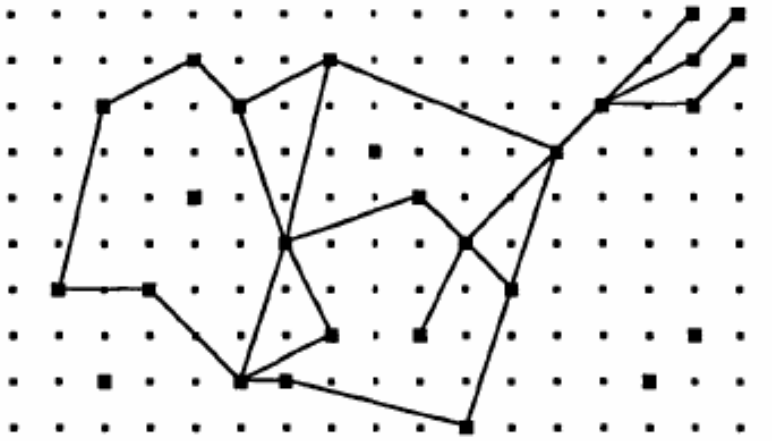
2.1 Konumsal Veri Yapıları ve Konumsal Cebir

Konumsal veri yapıları ya da konumsal cebirler, nokta, doğru ve bölgenin ana yapılarını ve aralarındaki ilişkileri yürütebilecek sistemlerdir. Konumsal cebirde en önemli örnek ROSE Cebiridir [24] . ROSE Cebiri **Realm** uzay yapısını temel alan bir konumsal veri tipi ve operasyonları sistemidir.

2.1.1 Realm:

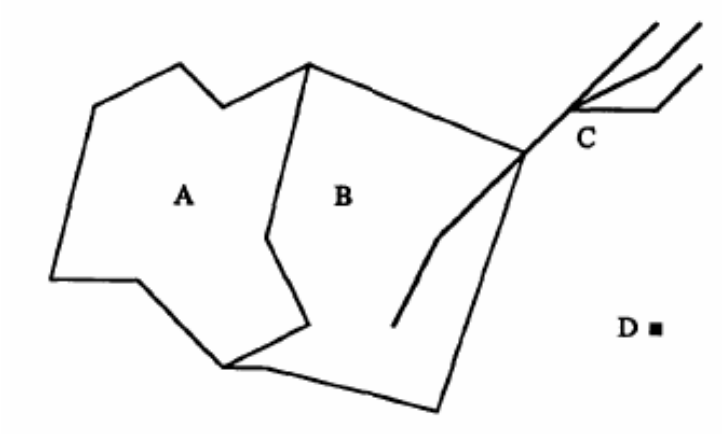
Bir realm ayrık bir alanın içinde noktalar kümesi ve kesişmeyen doğru parçacıklarından oluşan Şekil (2.2) 8'deki gibi bir ızgaradır [27] . Realm için bazı kurallar bulunmaktadır:

1. Her nokta ya da her doğrunun uç noktaları bir ızgara noktasıdır.
2. Her doğrunun bitiş noktası yine realm de bir noktadır.
3. Hiçbir realm noktası doğru parçasının üzerinde durmaz.
4. Hiçbir realm parçası uç noktaları dışında kesişmez.



Şekil (2.2) 8 Bir Realm örneği.

Konumsal veri tiplerinin değerleri realm deki gösterimlerinden oluşur. Şekil (2.3) 9, Şekil (2.2) 8 'deki tanımlanabilecek bazı nesneler göstermektedir.



Şekil (2.3) 9 Şekil 9'daki realm örneğinden tanımlanan nesneler.

Realm'in altındaki ızgara basit olarak sayıların bilgisayar hafızasında sonlu sayıda temsilleri olduğu gerçeğinden doğmuştur. Pratikte bu temsiller sabit uzunlukta olur ve programlama dillerinde bulunan INTEGER veya REAL veri türlerine (ya da daha özel olarak, sayı sistemlerine) karşılık gelmektedir. Tabi ki sonuç Şekil 10'da örneklendiğinden çok daha iyi olacaktır.

Uzaysal veri türlerinin temelini oluşturan realm kavramı şu amaçlara hizmet etmektedir:

- Uzaysal nesnelerin aralarındaki ilişkilerde tutarlılığı sağlar. Örneğin, A ve B ülkelerinin sınırlarının ortak kısımları her iki nesne içinde tamamıyla aynıdır.

- Realm'in üzerindeki uzaysal verilerin aralarındaki işlemlerde kapalılık özelliklerini garanti eder. Örneğin, bölüm B'nin C çizgisi ile kesişimi (B ülkesinin içinde uzanan C nehrinin parçası) de yine realm'e dayalı çizgi değeridir.

- Sorgulama sürecindeki geometrik hesaplamayı sayısal doğruluk ve sağlamlık problemlerinden korur. Çünkü bu tür problemler, aslında normalde ızgara üzerinde yer almayan doğru parçalarının kesişim noktalarının hesaplanmasından doğar. Realm'e dayalı konumsal veri yapıları ile sorgu sürecinde hesaplanan yeni kesişim noktaları yoktur. Onun yerine, sayısal problemler realm düzeyinin altında, ne zaman realm'e güncelleştirmeler yapılırsa o zaman ele alınır

- Buna ek olarak, realm'i temsil eden bir veri yapısı veri tabanına dizin olarak kullanılabilir. Uygulama kavramı realm 'deki her noktanın ve parçanın veritabanında uzaysal nitelik değerleri tanımlanan mantıksal gösterge ile ilişkili listesi olduğunu varsayar.

2.1.2 ROSE Katmanları

ROSE Cebiri'nin temel aldığı 4 ana tanımlama katmanı bulunmaktadır bunlar:

Katman 1 – Sağlam geometrik temeller: Altı çizili (ör. Intersect)

Katman 2 – Realm'ler, Realm tabanlı birinciller: Altı çizili italik (ör.*area-disjoint*)

Katman 3 – Konumsal cebir birincilleri: Kalın italik (ör. ***area-disjoint***)

Katman 4 – ROSE operasyonları: Kalın (ör. **inside**)

Aşağıda bu katmanların içerdikleri nesne ve operasyonlar sıralanmıştır.

Sağlam geometrik birinciller	Nesneler: N-point N-segment Operasyonlar: =, meet, overlap, intersect, disjoint, on, in, intersection,parallel, aligned
Realm'ler, Realm tabanlı birinciller	Nesneler: R-point, R-segment; R-cycle, R-face, R-unit, R-block Operasyonlar: on, in, out, (area-)inside, edge-inside, vertex-inside, area-disjoint, edge-disjoint, (vertex-)disjoint, adjacent, meet, encloses, intersect, dist, area
Konumsal cebir birincilleri	Nesneler: *points*, *lines*, *regions* Operasyonlar: ***union***, ***intersection***,***difference***, ***(area-)inside***, ***edge-inside***, ***vertex-inside***, ***area-disjoint***, ***(vertex-)disjoint***, ***adjacent***, ***meet***,

	intersect, ***encloses, on_border_of***, ***border_in_common***
ROSE operasyonları	Nesneler: *points*, *lines*, *regions* Operasyonlar: =, ≠, **inside , edge_inside, vertex_inside, area_disjoint, edge_disjoint, disjoint, intersects, meets, adjacent, encloses, on_border_of, border_in_common, intersection, blus, minus, common_border, vertices, contour, interior, count, dist, diameter, length, area, perimeter, sum, closest, decompose, overlay, fusion**

Tablo 1 ROSE Katman ve Operasyonları

2.1.3 Second Order Signiture

Rose Cebirinin tip sistemi **second-order signiture** [24] gösterimini temel almaktadır ki bu sistem çok çeşitli operasyonları tip kümeleri ile ölçmeye yarar [30] .

Bu tip sisteminin terimleri bir tip koleksiyonu tanımlar ki bu tip sistemidir. Basit bir örnek aşağıda verilmektedir. Her doğru bir operatörler grubunu tanımlar (bu durumda tip oluşturucularıdırlar).

kind DATA, GEO, SET

Tip oluşturucuları

	→DATA	int, real, bool
	→GEO	points, lines, regions
GEO	→SET	set

Burada int, set, … Tip oluşturucularıdır ki onlarda genellikle bir ya da daha fazla argüman türleri ve bir sonuç türü olur. Sıfır argüman türü olan tip oluşturucularına sabit tipler denir. Yukarıdaki örnekte, set dışındaki oluşturucular sabit tiplerdir. Bu tip sistemi sonuç türü olarak sınıflandırılabilir. Örneğin, GEO türünün tam olarak üç tipi bulunmaktadır. SET türünün tipleri set(points), set(lines) ve set(regions) olarak karşımıza çıkmaktadır.

2.1.4 ROSE Cebiri

Rose cebiri üç veri yapısı önermektedir bunlar; noktalar, doğrular ve alanlar'dır ki bunlar realm tabanlı veri yapılarıdır. Bu değerleri anlatabilmek için R-Block ve R-Face tanılanmalıdır. Verilen bir realm R'de bir R-Block R'nin bağlı doğru parçaları kümesidir. Bir R-Face ise esasen R parçacıkları ile ifade edilebilen delikli bir poligondur. Bu durumda, bir nokta tipi değeri R-Points kümesidir, bir doğru tipi değeri ayrık R-Block kümesidir ve bir bölge tipi değeri de kenarları ayrık (köşeleri değebilir) R-Face tipidir denebilir.

Rose Cebiri Realm tabanlı bir cebirdir çünkü veri tipleri realms üzerinde tanımlanmıştır. Rose Cebirini tanımlamak için second-order signiture gösteriminde points, lines ve regions tipleri tanımlanmıştır. Bundan sonra cebir, tipler için taşıyıcı kümeler ve operasyonlar için fonksiyonlardan oluşur denebilir.

Rose Cebirinin tip sistemi şu şekilde özetlenebilir.

kinds INDENT, DATA, EXT, GEO, OBJ, SET

type constructors

→INDENT	indent
→DATA	int,real,bool,…
→EXT	lines,regions
→GEO	points,lines,regions

OBJ →SET set

DATA türü standart veri tiplerini ifade etmektedir. EXT doğrular ve bölgeleri içine almaktadır ki EXT noktalar için uygun olmayan operasyonları tanımlamakta kullanılır.

Esasen iki tip kümesi bulunmaktadır bunlar EXT = (doğrular, bölgeler} ve GEO = {noktalar, doğrular, bölgeler}.

ROSE Cebirinde dört operasyon sınıfı tanımlanmıştır bunlar;

1) Topolojik ilişkileri ifade eden konumsal önermeler:

∀ geo in GEO. ∀ extl, ext2 in EXT. ∀ bölge içindeki alanlar *ayrık-alanlar*

geo × *bölgeler* → *bool* **içerde**
ext1 × *ext2* → *bool* **kesişir, buluşur**
alan x *alan* → *bool* **komşu, kapatır**

Burada basit ya da karmaşık olan geometrik nesnelerin aralarındaki ilişkiler ifade edilmektedir. Topolojik nesneler yeri ya da şekli değişmeyen nesnelerdir.

2) Atomik konumsal veri tip değerlerini döndüren operasyonlar:

∀ geo in GEO.
doğrular x *doğrular* → *noktalar* **kesişim**
bölgeler x *bölgeler* → *bölgeler* **kesişim**
geo x *geo* → *geo* **artı, eksi**
bölgeler → *doğrular* **çevre çizgisi**

Burada artı eksi birleşim ve farkı temsil etmektedir. Bu fark aynı tipteki nesneler için geçerlidir.

3) Sayı döndüren konumsal operatörler:

∀ geol x geo2 in GEO.
geol × *geo2* → *reel sayı* **uzaklık**

bölgeler → *reel sayı* **çevre uzunluğu, alan**

4) Nesne kümelerinde konumsal operasyonlar:

∀ *obj* OBJ içinde. ∀ geo, geol, geo2 GEO içinde.
set(*obj*) X (obj → *geo*) → *geo* **toplam**
set(*obj*) X (obj → *geo1*) x *geo2* → *set*(*obj*) **en yakın**

Burada toplam konumsal toplam (aggregate) fonksiyonunu temsil etmektedir. İçerisine konumsal özellikleri ile bir nesne kümesi alır ve tüm özelliklerin geometrik birleşimini döndürür. Örneğin bir ülkenin alanını hesaplamak için iller birleştirilebilir. En yakın operatörü ise bir nesne kümesinin içerisindeki konumsal özellikleri başka bir konumsal nesneye en yakın özelliktekileri bulur.

Güting ve Schneider a göre (1993) [25] konumsal veri tipleri ve operasyonları hakkında bazı önemli kavramları bulunmaktadır.

- Genişletilebilirlik: Genel olarak konumsal veri tipleri ve operasyonları uygulamaya özel olmaktadır. Fakat bu veri tipleri ve operasyonları daha sonra genişletilebilmeli yeni operasyonlar ve veri tipleri eklenebilir olarak tanımlanmalıdır.

- Eksiksizlik: Esasen bir alanda eksiksiz olarak iş gören operasyonlar kümesi olup olmadığını gösteren resmi bir ölçüt olup olmadığı bir sorudur.

- Bir veya daha çok tip? Acaba gerçekten ayrık veri tipleri olması gerekmekte midir (Örneğin nokta, doğru, bölge)? Bazı araştırmacılar sadece bir geometrik veri tipi olmasını ve bu tipin o andaki değeri bu tiplerden herhangi biri olabilir önerisi getirmişlerdir (Gargano et al., 1991 [21] ; Larue et al., 1993 [32]). Bu aslında geleneksel bilgisayar problemlerinden birine çok benzemektedir, tam sayı ve reel sayı diye farklı türler bulunmalı mıdır ya da sadece sayı adında bir değişken mi olmalıdır? Tek tipin bir avantajı en yakınlık işlemlerini kolaylıkla hesaplayabilmektir. Fakat birden çok tip daha keskin sonuçlar üretmeye yarar.

- Operasyon Kümesi: Bir konumsal cebir sadece atomik konumsal veri tipleri arasındaki operasyonları değil, konumsal olarak ilişkili nesne kümeleri arasındaki

operasyonları da tanımlamalıdır. Mesela iki parselin örtüşmesi ya da bir nesneye en yakın nesnenin bulunması.

Bununla birlikte Eliseo Clementini ve Paolino Di Felice "A global framework for qualitative shape description" [7] adlı çalışmasında konumsal operatörleri üç ana başlık altında toplamıştır.

- Topolojik Operatörler: Topolojik operatörler ile bağlantıları, bileşen sayılarını ve topolojik ilişkileri (ki bunlara örnek vermek gerekirse iki nesne kesişiyor mu, kesişiyorsa nasıl kesişiyorlar) gibi önermeleri tanımlayabiliriz.

- İzdüşümsel Operatörler: İz düşümsel operatörler ile iç bükeylik ya da dış bükeylik gibi önermeleri tanımlarken aynı zamanda "*iç bükey içinde mi"* gibi konumsal ilişkileri tanımlayabiliriz.

- Metrik Operatörler: Metrik operatörler ile mesafe ve yön ilişkilerini tanımlayabiliriz. Bununla birlikte simetri ya da yoğunluk özellikleri de metrik operatörler ile tanımlanabilir.

Konumsal operatörleri genel bir standarda yerleştirebilmek için bazı çalışmalar ve tanımlamalar yapılmıştır. Yani bazı araştırmacılara göre konumsal operatör tanımlanırken bu operatörlerin bir takım kıstaslara uygun olması gerekmektedir.

Yine konumsal operatörlerin olması gereken özelliklerini Eliseo Clementini ve Paolino Di Felice tarafından tanımlanmıştır [5]

- Küçük operatörler kümesi: Kullanıcının öğrenme süresini azaltmak için operatörler sayı bakımından az olmalıdır. Örneğin topolojik açıdan tüm geometrik operasyonların sorgulama diline gömülmesi gereksiz ve zor bir iştir. Mesela, Egenhofer ve David M. Mark (95) [16] tarafından önerilmiş olan 9-kesişim metodunda iki boyutlu nesnelerin aralarında ayrık 56 ilişki türü bulunmaktadır. Ama gerçekçi bir bakış açısıyla 56 operatörün sorgulama dili açısından kullanıcı için çok fazla olduğu söylenebilir.

- Açıklayıcı olma: Genellikle konumsal sorgular konumsal olmayan sorgulara göre geometri nedeniyle daha karmaşıktır. Konumsal operatörlerin ya da konumsal sorgulama dilinin açıklayıcı olması öncelikli çalışma konusudur.

- Tutarlılık: Operatörler tutarsız sonuçlar getirmemelidir. Esasen bu formal yaklaşımlar ile başarılmıştır. İlişkilerin teorik özellikleri olan tamamlık ve karşıklı dışlamadan yararlanılarak tutarlılıkları ispatlanmıştır.

- Genelleştirme: Operatörler soyut geometrik veri yapıları seviyesinde tanımlanmalı ve uygulama bağımsız olmalıdırlar. Son yıllarda geometrik veri modelini destekleyecek konumsal veri modeli üstüne birçok çalışma yapılmıştır (Ralf H. Guting 94 [29] , Eliseo Clementini 1996 [8]). Bu veri modellerinde gerçek objeler geometrik veri tipleri olarak görülebilirler.

- Hiyerarşik yapı: Sorgulama dili kullanıcıya hiyerarşik operatörler kümesi sunmalıdır ki kullanıcı sorguyu istenilen geometrik detay seviyesinde tutabilsin. Yüksek seviyeli operatörler hızlı görüntüleme için, daha detaylı operatörler ise sonucu kısıtlamak için kullanılabilir.

- Kesin olmayan eşleme: Sorgulama dili belirsizliklerden sorumlu operatörleri kapsamalıdır. Sorguda kullanıcının belirlediği kıstaslara göre konumsal ölçüm benzerliklerini bulmakta kullanılmak üzere kullanılacak operatörler olarak da tanımlanabilir. Kesin olmayan eşleme özellikle çoklu ortan veritabanlarında kullanılmaktadır.

- Kullanıcı İçin Dil bilimsel ve Kavramsal Benzerlik: Konumsal operatörler kabul görmüş konumsal terimleri ve onların anlamsal karşılıklarını karşılamalıdır. Konumsal bir model oluşturabilmek ve onu insanların daha kolay kullanabilmeleri sağlamak, operatörlerin dile ve kavramlara daha yakın olmasına bağlıdır.

- Niteleyicilik: Operatörler kullanıcının sorguları nitel verilerle oluşturabilmelerini sağlamalıdır. Var olan sorgu dileri kesin sonuçlar döndürecek sorgulara cevap vermede çok iyidirler. Fakat günlük yaşamda kullanıcının konuşma

diline cevap vermekte yetersizdirler. Mesela Antalya hava alanına yakın olan otelleri getir denilememektedir.

- Belirsizlik Desteği: Operatörler sınırları kesin olan nesnelere destek vermekte de olsalar, daha geniş sınırları olan nesnelere de destek vermektedirler. Kullanıcının aşırı yüklenmesini sınırlayabilmek için önerilen, geniş sınırları olan nesnelere tek bir işlem için oluşturulan operatörlerin sayısı kısıtlanmalıdır.

2.2 Konumsal İlişkiler ve İlişkisel Operasyonlar

Konumsal Cebir'in bize sunduğu en önemli kavram konumsal ilişkilerdir. Mesela verilen bir ilişkinin içindeki tüm nesneler istenebilir (ör: Fare ile çizilen bir pencerenin içindeki tüm nesneler). Bazı araştırmacılar bu ilişkileri sınıflara ayırmışlardır (Pullar and Egenhofer, 1988 [13] ; Egenhofer, 1989 [15] ; Worboys, 1992 [39]) temel olarak bu sınıflandırma ortak bir çatı altında toplanabilir:

- Topolojik ilişkiler; kesişen, içinde, örtüşen gibi ilişkiler bu sınıfa girerler.
- Yön ilişkileri; yukarı, aşağı, kuzey-doğu, güney-batı gibi ilişkiler.
- Metrik ilişkiler; mesafe> 100km bu ilişki sınıfına örnek olabilir.

Geometrik nesneler arasındaki ilişkileri tanımlamak esasen aralarındaki geçerli operasyonları tanımlamak anlamına gelmektedir. İlişkiler Operatörler iki geometri arasında belirtilen bir topolojik uzamsal ilişkinin varlığını test etmek için kullanılır. İki geometrik nesne arasındaki topolojik uzamsal ilişki literatürde geniş kapsamlı olarak çalışılmıştır ([16] , [18] , [4] , [6] , [8]) . İki geometrinin karşılaştırılmasını içeren bu temel yaklaşım, iki geometrinin İçleri, Sınırları ve Dışları arasındaki kesişmelerin çiftli testlerini gerçekleştirmek ve sonuçta elde edilen "kesişim" matrisindeki girdilere dayalı iki geometri arasındaki ilişkiyi sınıflandırmaktır.

İki geometrinin aralarındaki ilişkileri iç ve sınır değerlerinin etkileşimi olarak değerlendirip aralarındaki ilişkileri sınıflandırmak için 4 Kesişim Modeli

tanımlanmıştır [18] . Model, girdi geometrilerinin dışını değerlendirecek şekilde genişletilmiş, bu da bir dokuz kesişme modeli ile sonuçlanmıştır [16] ve çiftli kesişmelerin sonuçlarının boyutu hakkında bilgileri içerecek şekilde tekrar genişletilmiş ve bu da boyutsal olarak genişleyen dokuz kesişme modeli ile sonuçlanmıştır [4] . Bu uzatımlar, modelin, delikli alanlar ve çoklu bileşenli çizgiler ve alanlar dâhil olmak üzere, noktalar, çizgiler ve alanlar arasındaki uzamsal ilişkileri ifade etmesine imkân tanır [8] .

2.2.1 Dört Kesişim Modeli

Dört Kesişim Modeli Egenhofer [18] tarafından iki geometrik nesne arasındaki kesişimleri değerlendirip ilişkileri sınıflandırmak için geliştirilmiştir. İki nesnenin iç ve sınır değerleri bu modelin temel aldığı nesne özellikleridir. A^0 A nesnesinin içini, ∂ A ise A nesnesinin sınırlarını temsil eder. Öyleyse iki nesne için 4 ilişki kümesi bulunmaktadır bunlar: $\partial A_1 \bigcap \partial A_2$, $\partial A_1 \bigcap A_2^0$, $A_1^0 \bigcap \partial A_2$ ve $A_1^0 \bigcap A_2^0$ dir. Bu durumda karşımıza Tablo 2'deki gibi $2^4 = 16$ değişik olasılık çıkmaktadır.

$\partial A_1 \bigcap \partial A_2$	$\partial A_1 \bigcap A_2^0$	$A_1^0 \bigcap \partial A_2$	$A_1^0 \bigcap A_2^0$	İlişki
Ø	Ø	Ø	Ø	A_1 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 'in içinde

Ø	≠Ø	≠Ø	Ø	
Ø	≠Ø	≠Ø	≠Ø	
≠Ø	Ø	Ø	Ø	A_1 A_2 'ye dokunur
≠Ø	Ø	Ø	≠Ø	A_1 A_2 eşittirler
≠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 4 Kesişim Modelindeki 16 değişik olasılık

Bu olasılıkların 8 tanesi geçersiz iki tanesi de simetriktir. Geriye altı ayrı ilişki kalmaktadır bunlar; *ayrık*, *içinde*, *değen*, *eşit*, *kapsar* ve *örtüşür* olarak isimlendirilmektedir.

2.2.2 Dokuz Kesişim Modeli

Max J. Egenhofer [16] tarafından geliştirilmiş bu modelin amacı iki boyutlu cisimlerin etkileşimlerini formalize etmektir. Bu model temel olarak 4-Kesişim

Modelinin, geometrik nesnelerin dışını (tümleyenini) ekleyerek genişletilmesidir. R iki A ve B nesnelerinin ikili topolojik ilişkisini temsil eder.

- A^0 A'nın içini temsil eder.
- ∂A A'nın sınırlarını temsil eder.
- A^- A'nın dışını temsil eder.
- B^0 B'nın içini temsil eder.
- ∂B B'nın sınırlarını temsil eder.
- B^- B'nın dışını temsil eder.

- $A^0 \cap B^0$ A ve B'nin iç kısımlarının kesişimini gösterir.
- $A^0 \cap \partial B$ A'nın içi ile B'nin sınırlarının kesişimini gösterir.
- $A^0 \cap B^-$ A'nın içi ile B'nin dışı arasındaki kesişimi gösterir.
- $\partial A \cap \partial B$ A ve B'nin sınırlarının kesişimlerini gösterir.
- $\partial A \cap B^0$ A'nın sınırlarıyla B'nin içinin kesişimini gösterir.
- $\partial A \cap B^-$ A'nın sınırlarıyla B'nin dışının kesişimini gösterir.
- $A^- \cap B^-$ A ve B nesnelerinin dışlarının kesişimini gösterir.
- $A^- \cap \partial B$ A'nın dışı ile B'nin sınırlarının kesişimini gösterir.
- $A^- \cap B^0$ A'nın dışı ile B'nin içinin kesişimini gösterir.

Bazen $A^0 \cap B^0$, $A^0 \cap \partial B$, $A^0 \cap B^-$ bu üç ifade A'nın dâhili kesişimleri olarak nitelendirilir. Ya da $\partial A \cap \partial B$, $A^0 \cap \partial B$, $A^- \cap \partial B$ bu üç ifade B'nin sınır kesişimleri olarak ifade edilir.

A ve B nesnelerinin arasındaki topolojik ilişkileri tanımlamak için bu dokuz kesişimin matrisi kullanılır (3x3).

$$R(A,B) = \begin{Bmatrix} A^0 \cap B^0 & A^0 \cap \partial B & A^0 \cap B^- \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^0 & A^- \cap \partial B & A^- \cap B^- \end{Bmatrix}$$

Her değişik 9-kesişim matrisi değişik bir topolojik ilişkiyi ifade eder. 9 Kesişimin içeriği boş(Ø) ya da boş değil($\neg Ø$) değerlerinden oluşur. Örneğin A bölgesinin B bölgesini kapsadığının boş/boş değil kesişimlerine göre ifadesi şöyledir.

$$R(A, B) = \begin{Bmatrix} A^0 \cap B^0 = \neg Ø & A^0 \cap \partial B = Ø & A^0 \cap B^- = Ø \\ \partial A \cap B^0 = \neg Ø & \partial A \cap \partial B = \neg Ø & \partial A \cap B^- = Ø \\ A^- \cap B^0 = \neg Ø & A^- \cap \partial B = \neg Ø & A^- \cap B^- = \neg Ø \end{Bmatrix}$$

ya da kısaca

$$R(A, B) = \begin{Bmatrix} \neg Ø & Ø & Ø \\ \neg Ø & \neg Ø & Ø \\ \neg Ø & \neg Ø & \neg Ø \end{Bmatrix}$$

Dokuz kesişimin sıralaması, yukardan aşağıya ve soldan sağa, daima 1) İç bölge 2) sınır 3) harici bölge şeklinde olacaktır.

Topolojik Özellikler:

İki nesne arasındaki topolojik özellikler dokuz kesişim modeli ile ifade edilebilir. Değerinin ne olduğu fark etmeyecek kesişimler (-) sembolü ile ifade edilir.

a_i Ve b_i A ve B nesnelerinden rasgele seçilmiş parçalar olsun.

• Eğer a_i b_i den ayrık ise, iki parçanın kesişimleri boş olmalıdır. Diğer 8 kesişim rasgele değerler olabilir. Örneğin; A'nın sınırları B'nin içinden ayrık ise A ve B'nin arasındaki 9 kesişim şu kalıpta olmalıdır.

$$R\{\text{Ø}, \neg\text{Ø}\}(A, B) = \begin{Bmatrix} - & - & - \\ \text{Ø} & - & - \\ - & - & - \end{Bmatrix}$$

• Eğer a_i b_i'yi kesiyorsa, iki parçanın kesişimleri boş olamamalıdır. Örneğin; A'nın içi B'nin sınırlarını kesiyorsa A ve B'nin arasındaki 9 kesişim şu kalıpta olmalıdır.

$$R\{\text{Ø}, \neg\text{Ø}\}(A, B) = \begin{Bmatrix} - & \neg\text{Ø} & - \\ - & - & - \\ - & - & - \end{Bmatrix}$$

• Eğer a_i b_i'nin alt kümesi ise ($\subseteq$), iki parçanın arasındaki kesişim boş olmamalı, bununla birlikte a_i ve diğer iki parçanın (b_k ve b_l) kesişimleri boş olmalıdır. Örneğin; A'nın sınırları B'nin içinin alt kümesi ise, A ve B'nin arasındaki 9 kesişim şu kalıpta olmalıdır.

$$R\{\text{Ø}, \neg\text{Ø}\}(A, B) = \begin{Bmatrix} - & - & - \\ \neg\text{Ø} & \text{Ø} & \text{Ø} \\ - & - & - \end{Bmatrix}$$

• Eğer a_i b_J ve b_k'nın alt kümesi ise ($j \neq k$), öyle ki $a_i \not\subset b_J$ ve $a_i \not\subset b_k$, a_i ve B'nin üçüncü bir parçası arasındaki kesişim boş iken, bu iki parça arasındaki kesişim boş olmamalıdır. Örneğin; eğer $\partial A \subseteq (\partial B \cup B^0)$ öyle ki $\partial A \not\subset \partial B$ ve $\partial A \not\subset B^0$ ise, Şekil (2.4) 10'deki A ve B'nin arasındaki 9 kesişim şu kalıpta olmalıdır.

Şekil (2.4) 10 A ve B alanlarının kesişimi

$$R\{\text{Ø}, \neg\text{Ø}\}(A, B) = \begin{Bmatrix} - & - & - \\ \neg\text{Ø} & \neg\text{Ø} & \text{Ø} \\ - & - & - \end{Bmatrix}$$

• Eğer iki nesne parçası a_i ve b_i çakışıyorlarsa aralarındaki kesişim boş olmamalıdır, bu arada diğer dört kesişim boş olmalıdır. Örneğin eğer A ve B'nin sınırları çakışıyorsa, A ve B'nin arasındaki 9 kesişim şu kalıpta olmalıdır.

$$R\{\text{Ø}, \neg\text{Ø}\}(A, B) = \begin{Bmatrix} - & \text{Ø} & - \\ \text{Ø} & \neg\text{Ø} & \text{Ø} \\ - & \text{Ø} & - \end{Bmatrix}$$

$\Re^2$'de ki 9-Kesişimler:

İki nesne arasında 2^9 olası topolojik ilişki bulunmaktadır, fakat bunların yalnız küçük bir kısmı uzayda anlamlıdır. Burada anlatılacak hangilerinin anlamlı, hangilerinin anlamsız olduğudur.

Kullanılacak teknik üç adımlı bir işlemdir:

• Var olmayan topolojik ilişkiler için oluşmuş topolojik durumları 9-kesişim boş/boş değil tekniği ile formal hale getirmek.

• Var olmayan ilişkilerin birleşim yolu ile var olan ilişkiler kümesini (512 adet) düşürmek.

• $\Re^2$'de ki kalan ve var olan ilişkileri doğrulamak.

İki Bölge Arasındaki İlişkiler

Koşul 1 İki hücrenin dış kısımlarının kesişmesi

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - & - \\ - & - & - \\ - & - & Ø \end{bmatrix}$$

İzleyen üç koşul bu konumsal veri modeline göre oluşmaktadır. İki bölgenin sınırları çakışmıyorsa, aralarında bazı dâhili ya da harici ilişki vardır. Bu şunu gösterir, eğer A'nın içi B'nin dışı ile kesişmiyorsa dâhili kısımlar kesişmelidir (Koşul 2), A'nın sınırı B'nin dışı ile kesişmemelidir (Koşul 3), ve A'nın içi B'nin sınırları ile kesişmemelidir (Koşul 4).

Koşul 2 Eğer her iki iç de ayrıksa, A'nın içi B'nin dışı ile kesişir, bunun terside geçerlidir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} Ø & - & Ø \\ - & - & - \\ - & - & - \end{bmatrix} V \begin{bmatrix} Ø & - & - \\ - & - & - \\ Ø & - & - \end{bmatrix}$$

Koşul 3 A'nın içi B'nin kapalı alanın alt kümesi ise, A'nın sınırları B'nin kapalı alanının alt kümesi olmalıdır, bunun terside geçerlidir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - & Ø \\ - & - & \neg Ø \\ - & - & - \end{bmatrix} V \begin{bmatrix} - & - & - \\ - & - & - \\ Ø & \neg Ø & - \end{bmatrix}$$

Koşul 4 Eğer A'nın içi B'nin sınırları ile kesişiyorsa, A'nın içi B'nin dışı ile de kesişir, bunun terside geçerlidir.

$$\begin{bmatrix} - & \neg Ø & Ø \\ & & \end{bmatrix} \quad \begin{bmatrix} - & - & - \\ & & \end{bmatrix}$$

$$R_{\{\emptyset,\neg\emptyset\}}(A,B) \neq \begin{matrix} - & - & - \\ - & - & - \end{matrix} \; V \; \begin{matrix} \neg\emptyset & - & - \\ \emptyset & - & - \end{matrix}$$

Boş olmayan sınırları olan bir hücrenin üç sınır kesişimleri boş olmaz. $\partial A = \neg\emptyset$, $\partial A \cap \Re^2 = \neg\emptyset$. $\partial B \cup B^0 \cup B- = \Re^2$ olduğu için $\partial A \cap (\partial B \cup B^0 \cup B-) = \neg\emptyset$, ki bu eğer en azından B'nin bir parçası A'nın sınırlarını kesiyorsa doğrudur.

Koşul 5 A'nın sınırları B'nin en azından bir parçasını keser, bunun terside doğrudur.

$$R_{\{\emptyset,\neg\emptyset\}}(A,B) \neq \begin{bmatrix} - & - & \emptyset \\ \emptyset & \emptyset & \emptyset \\ - & - & - \end{bmatrix} V \begin{bmatrix} - & \emptyset & - \\ - & \emptyset & - \\ - & \emptyset & - \end{bmatrix}$$

Bir bölgenin sınırları iç bölgesini dış bölgesinden ayırdığı için, dış bölgeden iç bölgeye giden tüm yollar sınırı keser (Jordan-Curve-Teorem) [E. Spanier, Algebraic Topology (McGraw-Hill Book Company, New York, 1966)] . Bu durum izleyen dört teoreme temel teşkil etmektedir.

Koşul 6 Eğer her iki iç bölge ayrık ise, A'nın sınırları B'nin iç bölgesini kesmez, bunun tersi de doğrudur.

$$R_{\{\emptyset,\neg\emptyset\}}(A,B) \neq \begin{bmatrix} \emptyset & \neg\emptyset & - \\ - & - & - \\ - & - & - \end{bmatrix} V \begin{bmatrix} \emptyset & - & - \\ \neg\emptyset & - & - \\ - & - & - \end{bmatrix}$$

Bir nesnenin hem içi hem de dışı ile kesişen her bağlı nesne parçası aynı zamanda o nesnenin sınırlarını da kesiyordur.

Koşul 7 Eğer A'nın içi B'nin hem içi ile hem de dışı ile kesişiyorsa, B'nin sınırlarını da kesiyor demektir, bunun terside doğrudur.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} \neg Ø & - & - \\ \neg Ø & - & - \\ \neg Ø & - & - \end{bmatrix} V \begin{bmatrix} \neg Ø & Ø & \neg Ø \\ - & - & - \\ - & - & - \end{bmatrix}$$

İki bölgenin sınırları iç içe değilse, en azından bir sınır diğer bölgenin dışı ile kesişir.

Koşul 8 Eğer iki sınır da kesişmiyorsa, birinin diğerinin sınırı diğerinin dış bölgesini keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - & - \\ - & Ø & Ø \\ - & Ø & - \end{bmatrix}$$

Eğer iki bölgenin iç tarafları ayrık ise, en az bir sınır diğerinin dış kısmını keser.

Koşul 9 Eğer her iki nesnenin iç bölgeleri birbirlerini kesmiyorsa, en az bir sınır diğerinin dış bölgesini keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} Ø & - & - \\ - & - & Ø \\ - & Ø & - \end{bmatrix}$$

Deliksiz Bölgeler için koşullar:

Koşul 1–9 nesnelerin delikli olup olmadığı fark etmeyen önermelerdir. Delikli bölgeler konumsal nesne sınıfları arasında topolojik ilişkileri daha kısıtlı olanlardır.

Deliksiz bölgelerin en önemli özelliği sınırlarının bağlı olasıdır. Olası durumlar Şekil (2.5) 11'de gösterilmiştir.

Koşul 10 Eğer her iki sınır ters içleri ile kesişiyorsa, sınırları da kesişiyor demektir.

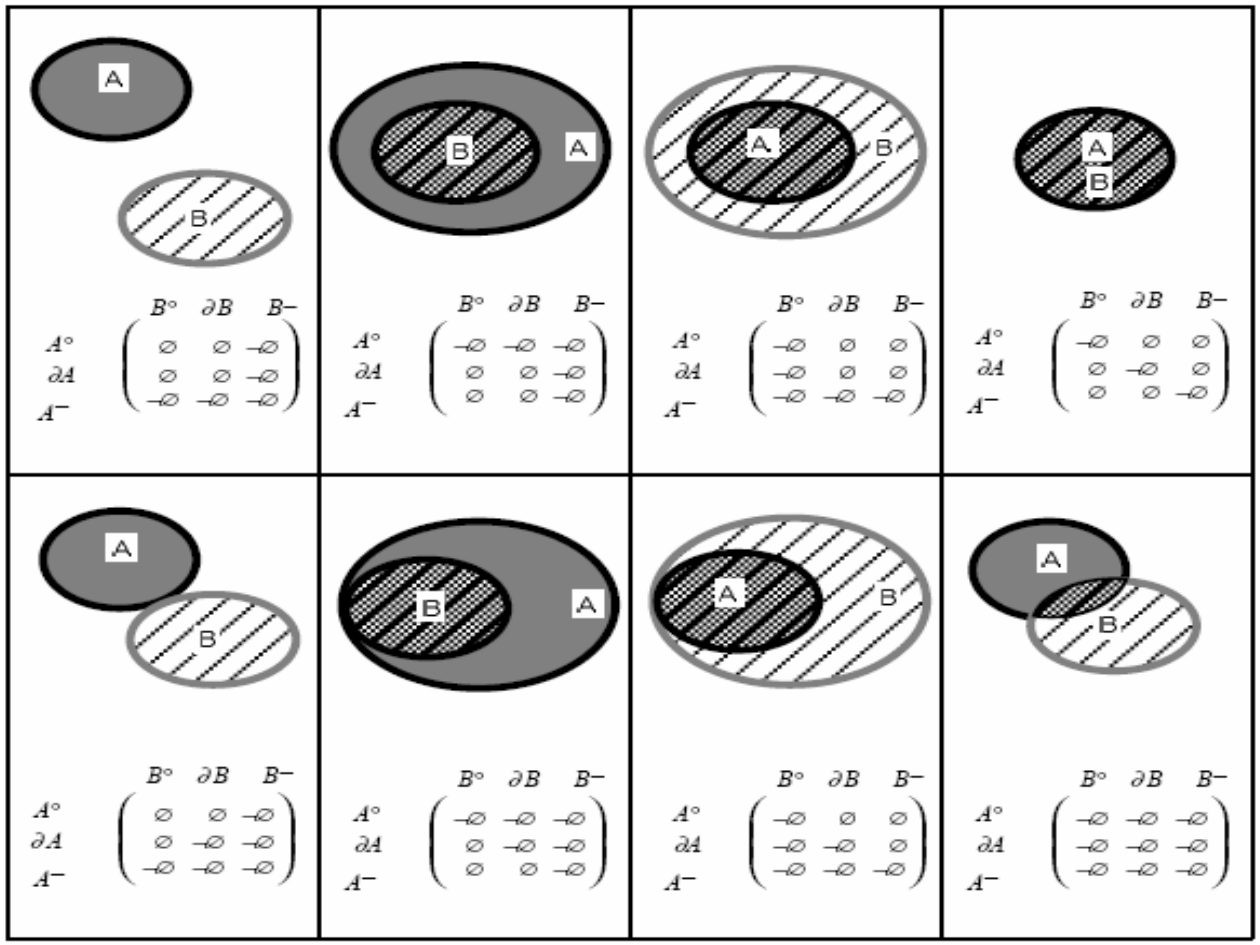
$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & \neg Ø & - \\ \neg Ø & Ø & - \\ - & - & - \end{bmatrix}$$

Koşul 11 Eğer A'nın içi B'nin dışını kesiyorsa, A'nın sınırları B'nin dışını da keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - & \neg Ø \\ - & - & Ø \\ - & - & - \end{bmatrix} V \begin{bmatrix} - & - & - \\ - & - & - \\ \neg Ø & Ø & - \end{bmatrix}$$

Koşul 12 Eğer iç bölgeler birbirleri ile kesişmiyorlarsa, A'nın sınırları B'nin dış kısmını kesiyor demektir, bunun terside doğrudur.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} Ø & - & - \\ - & - & - \\ - & Ø & - \end{bmatrix} V \begin{bmatrix} Ø & - & - \\ - & - & Ø \\ - & - & - \end{bmatrix}$$



Şekil (2.5) 11 Bağlı sınırları olan iki bölgenin aralarındaki 8 ilişki [16]

İki doğru için koşullar:

Doğrular boş olmayan hücreler ve sınırlardır, bu nedenle koşul 1–5 doğrulara uygundur. Eğer bir doğrunun dışı başka bir doğrunun sınırı ile kesişiyorsa, doğrunun dışı zaten diğer doğrunun içi ile kesişir. Özet olarak Şekil (2.6) 12 ve Şekil (2.7) 13'de basit doğrular için, Şekil (2.8) 14 ve Şekil (2.9) 15'da ise basit olmayan doğrular için ilişkiler verilmiştir.

Koşul 13 Eğer A'nın kapalı alanı B'nin içinin alt kümesi ise, ya A'nın dışı B'nin sınırları ve içi ile kesişir, ya da bu durumlardan hiçbiri olmaz, bunun terside geçerlidir.

$$R_{\{\emptyset,\neg\emptyset\}}(A,B) \neq \begin{pmatrix} - & - & \emptyset \\ - & - & - \\ \emptyset & \neg\emptyset & - \end{pmatrix} \text{V} \begin{pmatrix} - & - & \emptyset \\ - & - & \neg\emptyset \\ \emptyset & - & - \end{pmatrix}$$
$$\text{V} \begin{pmatrix} - & - & \emptyset \\ - & - & - \\ \neg\emptyset & \emptyset & - \end{pmatrix} \text{V} \begin{pmatrix} - & - & \neg\emptyset \\ - & - & \emptyset \\ \emptyset & - & - \end{pmatrix}$$

Basit Doğru Koşulları:

Eğer iki doğruda basit doğrular ise iki sınır da sadece iki noktadan oluşuyor demektir. Bu nedenle basit bir doğru, basit olan başka bir doğru parçasının sadece bir noktasını kesebilir. Bu durum izleyen koşula temel teşkil eder.

Koşul 14 Her iki sınır da en fazla iki karşıt parçayı kesebilir.

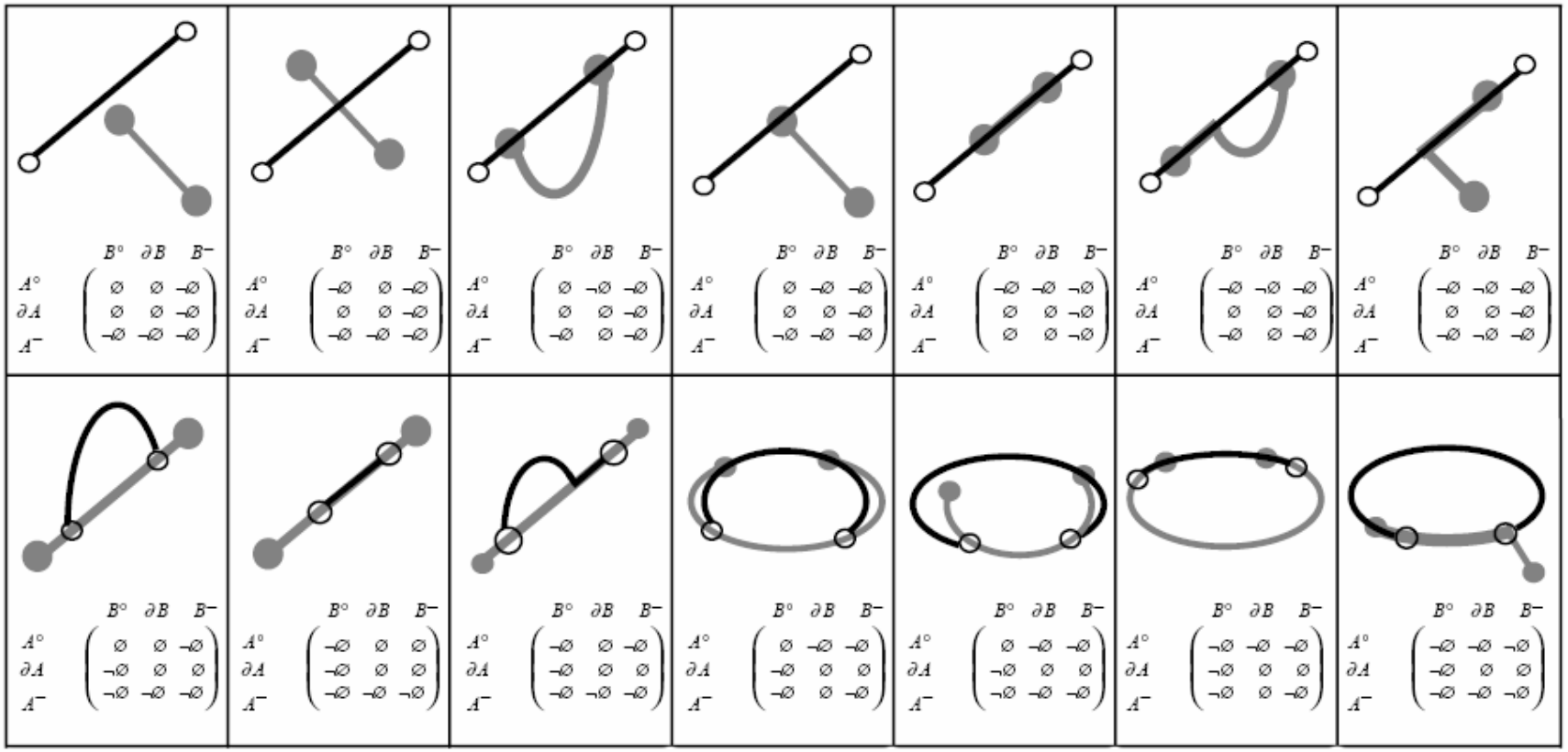
$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & \neg Ø & - \\ - & \neg Ø & - \\ - & \neg Ø & - \end{bmatrix} V \begin{bmatrix} - & - & - \\ \neg Ø & \neg Ø & \neg Ø \\ - & - & - \end{bmatrix}$$

Koşul 15 Eğer A'nın sınırı B'nin sınırının alt kümesi ise, iki sınır çakışıyor demektir, bunun terside doğrudur.

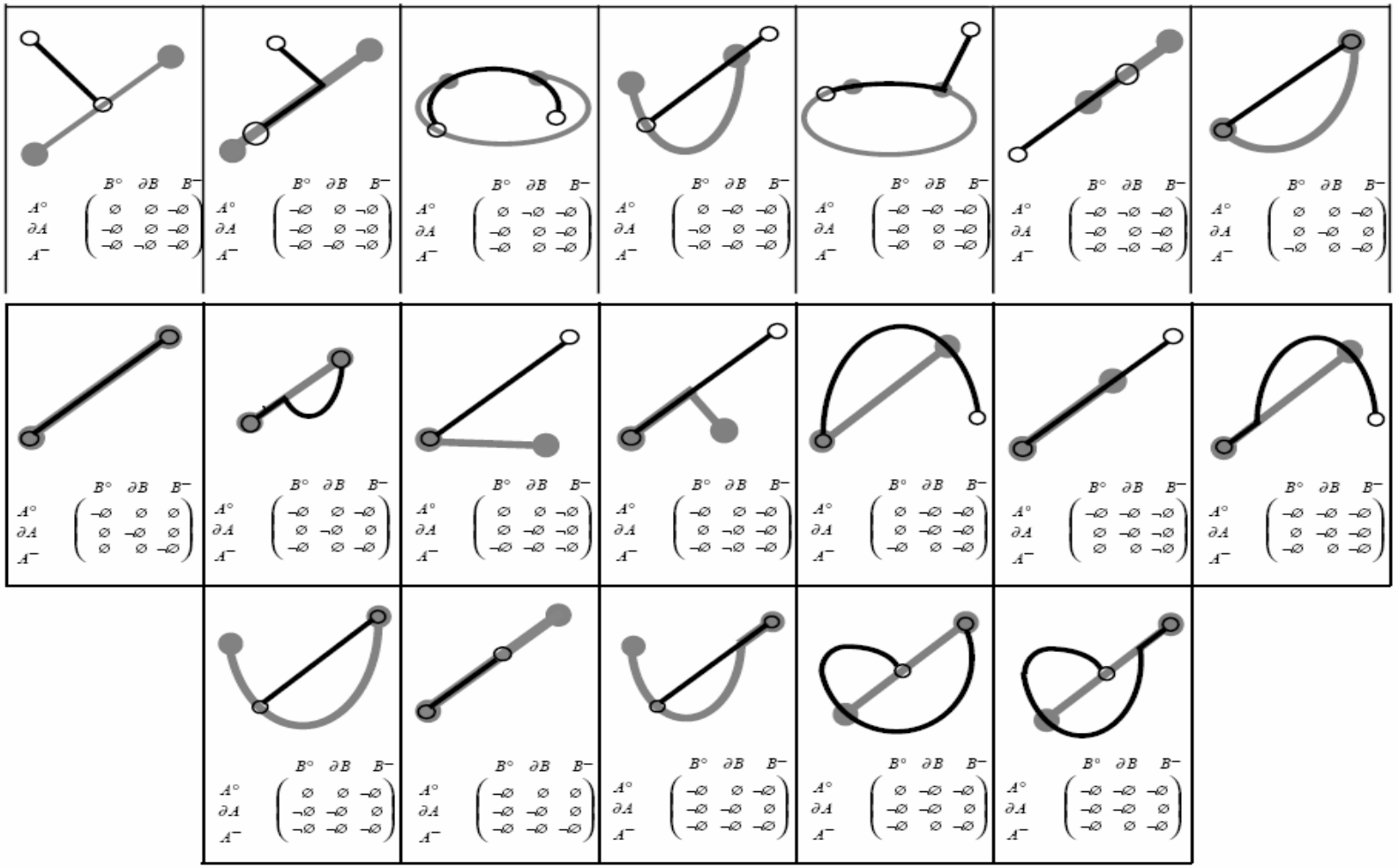
$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & \neg Ø & - \\ Ø & \neg Ø & Ø \\ - & - & - \end{bmatrix} V \begin{bmatrix} - & - & - \\ Ø & \neg Ø & Ø \\ - & \neg Ø & - \end{bmatrix}$$

$$\begin{bmatrix} - & Ø & - \\ \neg Ø & \neg Ø & - \\ - & Ø & - \end{bmatrix} V \begin{bmatrix} - & Ø & - \\ - & \neg Ø & \neg Ø \\ - & Ø & - \end{bmatrix}$$

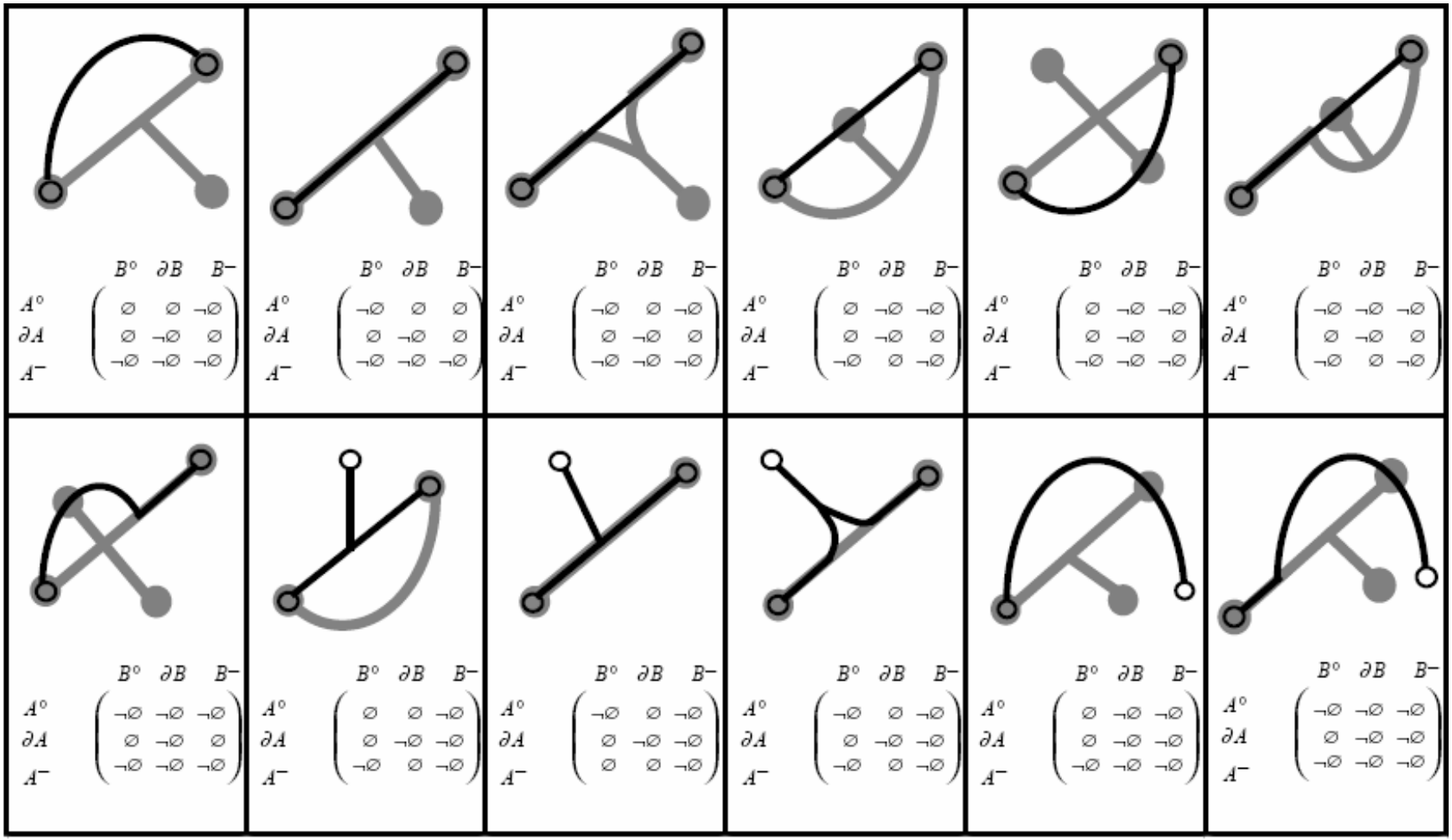
İki doğru arasında toplam 57 ilişki bulunmaktadır, bunlardan 33 ü basit doğrular ile gerçekleşebilir. 24 ilişki ise karmaşık doğrular ile gerçekleşir.



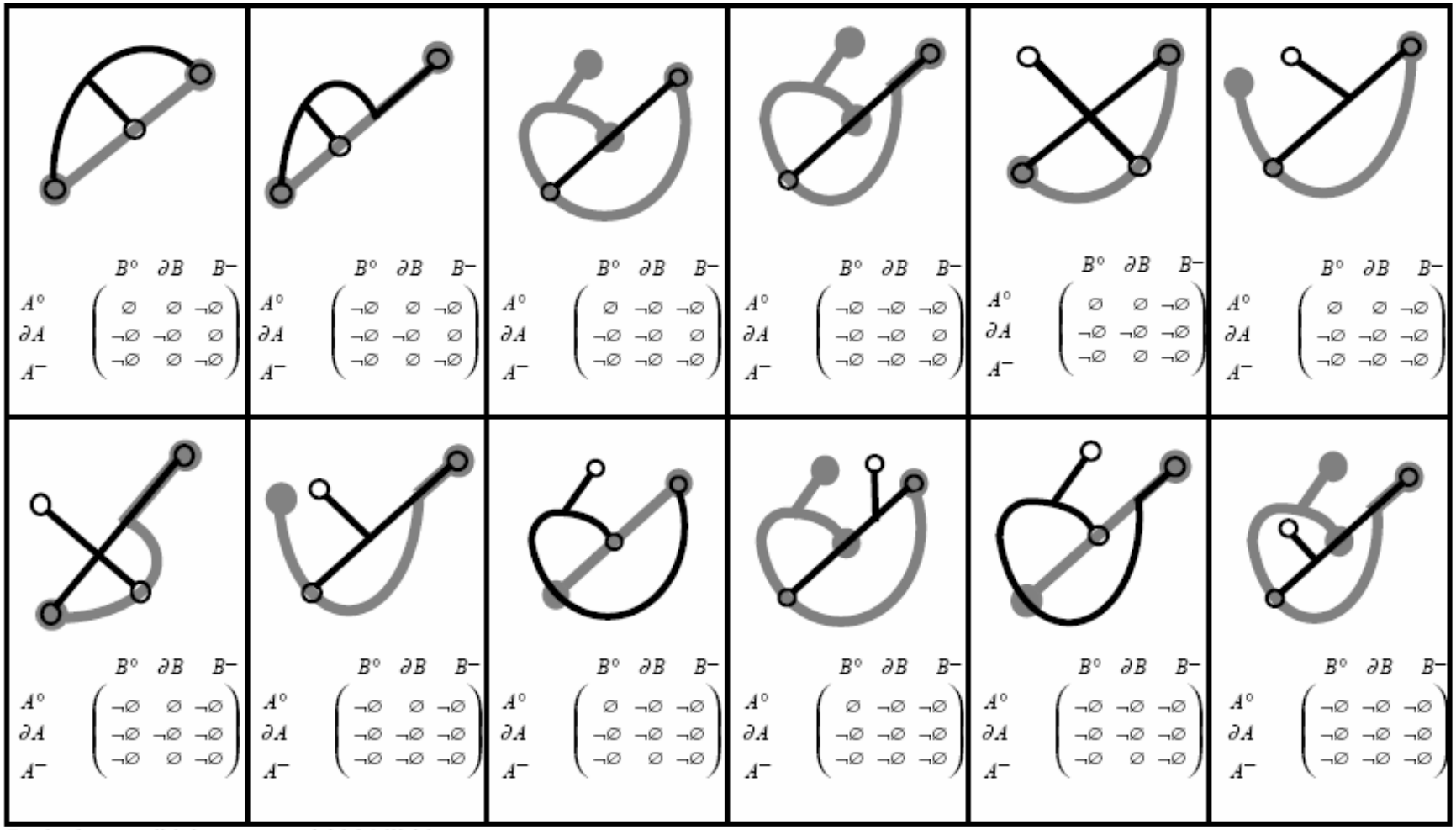
Şekil (2.6) 12 İki basit doğru arasında gerçekleşebilecek 33 ilişki [16]



Şekil (2.7) 13 İki basit doğru arasında gerçekleşebilecek 33 ilişki (devam) [16]



Şekil (2.8) 14 Basit olmayan iki doğru arasındaki 24 ilişki [16]



Basit olmayan iki doğru arasındaki 24 ilişki

Şekil (2.9) 15 Basit olmayan iki doğru arasındaki 24 ilişki (devam) [16]

Bir bölge ve bir doğru için koşullar:

Şekil (2.10) 16 ve Şekil (2.11) 17'de bir bölge ve bir doğru için olabilecek tüm doğrular verilmiştir.

Koşul 16 Bir A bölgesi daima bir B doğrusunun dışı ile kesişir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{pmatrix} - & - & Ø \\ - & - & - \\ - & - & - \end{pmatrix}$$

Bir doğrunun sınırları boş olamaz ve döngü oluşturamaz ve bir bölgenin sınırı kapalıdır tanımlarından yola çıkarak izleyen koşul tanımlanabilir.

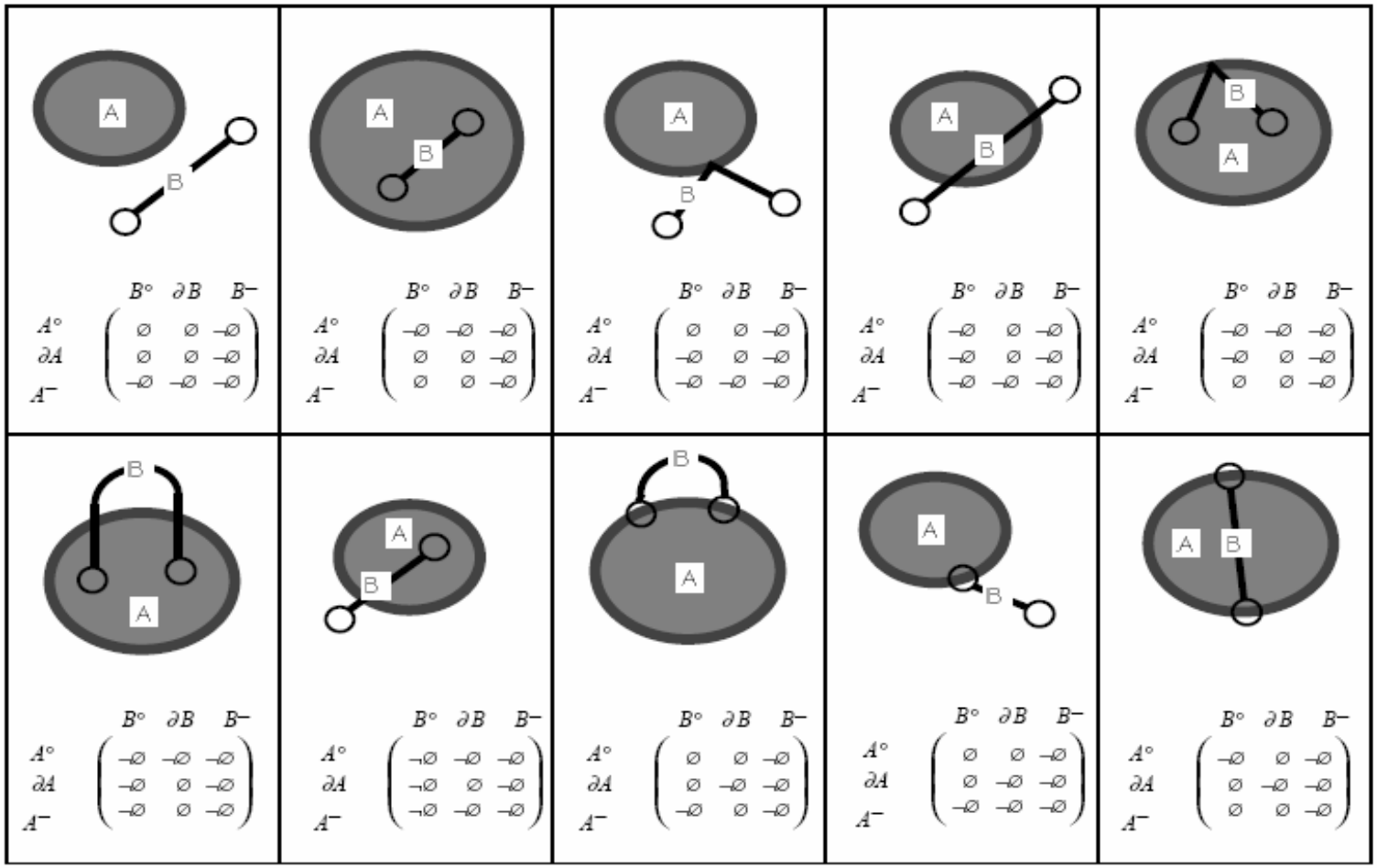
Koşul 17 A bölgesinin sınırları daima B doğrusunun dışını keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{pmatrix} - & - & - \\ - & - & Ø \\ - & - & - \end{pmatrix}$$

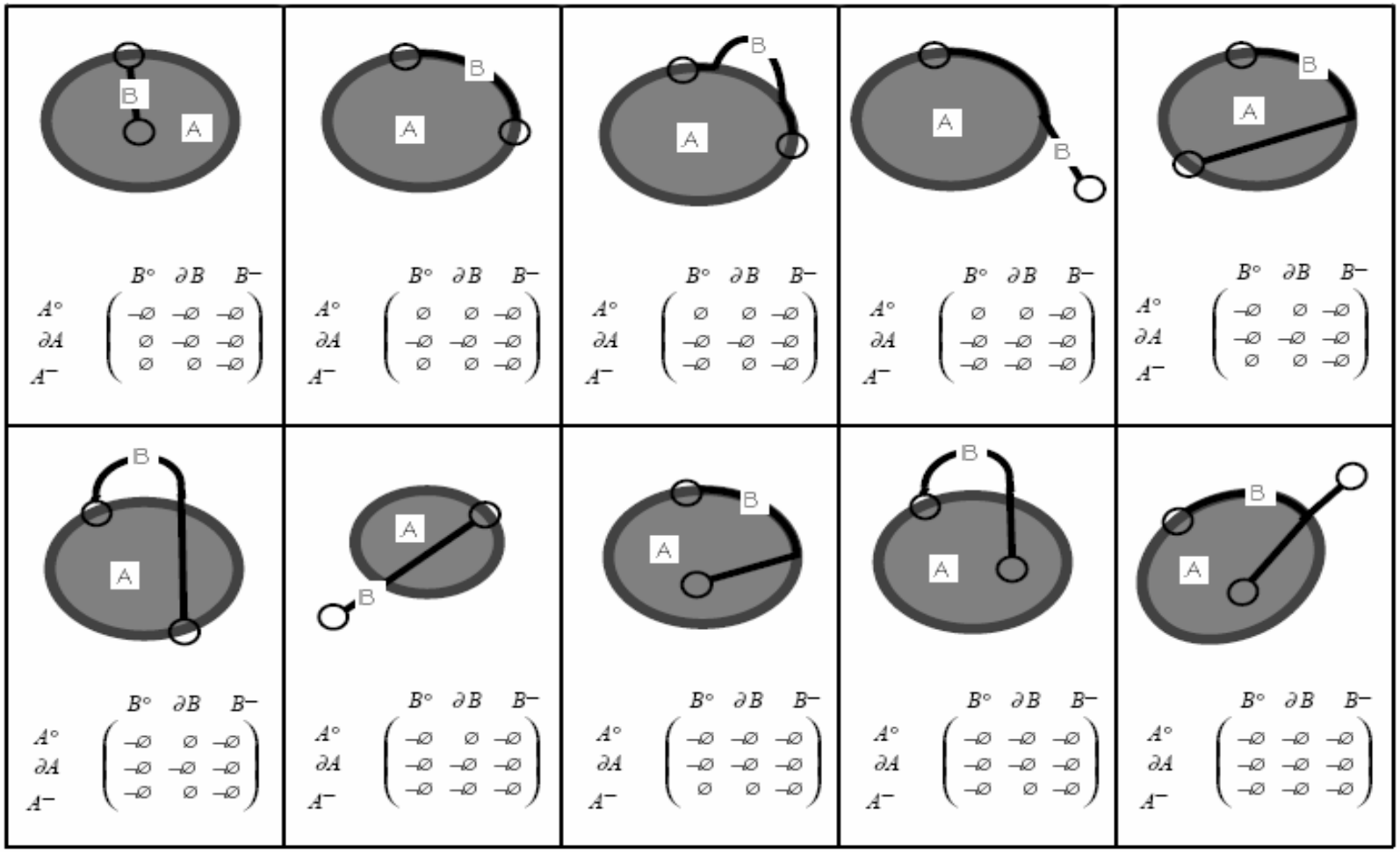
Bir doğrunun içi boş olamaz bu durumdan izleyen koşul ortaya çıkar.

Koşul 18 B doğrusunun içi mutlaka, B bölgesinin sınırlarını, ya da içini, ya da dışını keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{pmatrix} Ø & - & - \\ Ø & - & - \\ Ø & - & - \end{pmatrix}$$



Şekil (2.10) 16 Bir doğru ve bir bölge arasındaki 20 geometrik ilişki [16]



Bir doğru ve bir bölge arasındaki 20 geometrik ilişki

Şekil (2.11) 17 doğru ve bir bölge arasındaki 20 geometrik ilişki (Devam) [16]

Nokta nesneleri ile ilişkiler:

Nokta olan nesnelerin sınırları boştur, bu nedenle elimizde altı adet kesişim kalmaktadır.

Koşul 19 B noktasının dışı nokta olmayan A nesnesinin içi, dışı ve sınırlarıyla kesişir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - \\ - & Ø \\ - & - \end{bmatrix} V \begin{bmatrix} - & Ø \\ - & - \\ - & - \end{bmatrix} V \begin{bmatrix} - & - \\ - & - \\ - & Ø \end{bmatrix}$$

Koşul 20 Bir noktanın içi başka bir nesnenin sadece bir kısmı ile kesişebilir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} \neg Ø & - \\ \neg Ø & - \\ - & - \end{bmatrix} V \begin{bmatrix} - & - \\ \neg Ø & - \\ \neg Ø & - \end{bmatrix} V \begin{bmatrix} \neg Ø & - \\ - & - \\ \neg Ø & - \end{bmatrix}$$

Koşul 21 Bir noktanın iç kısmı başka bir nesnenin üç bölgesinden birinin alt kümesidir (iç, dış, sınırlar)

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} Ø & - \\ Ø & - \\ Ø & - \end{bmatrix}$$

Burada daha karmaşık olan iki nokta arasındaki ilişkilerdir. Her iki nesnenin de sınırları boş olduğundan sadece dört geçerli kesişim bulunmaktadır.

Koşul 22 Her iki nokta nesnesinin de dış kısımları mutlaka kesişir.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} - & - \\ - & Ø \end{bmatrix}$$

Noktalar atomik nesneler olduğu için diğer bir hücrenin sadece bir kısmını kesebilirler. Diğer taraftan noktalar boş olmayan nesneler olduğundan, en azından hücrenin bir kısmını keserler.

Koşul 23 Bir noktanın içi diğer bir nesnenin bir parçasını keser.

$$R_{\{Ø,\neg Ø\}}(A,B) \neq \begin{bmatrix} Ø & Ø \\ - & - \end{bmatrix} V \begin{bmatrix} Ø & - \\ Ø & - \end{bmatrix} V \begin{bmatrix} \neg Ø & \neg Ø \\ - & - \end{bmatrix} V \begin{bmatrix} \neg Ø & - \\ \neg Ø & - \end{bmatrix}$$

2.2.3 Boyutsal olarak Genişletilmiş Dokuz Kesişme Modeli (DE-9IM)

Boyutsal olarak uzatılmış Dokuz Kesişim Modeli Clementini ve Felice [4] tarafından geliştirilmiş ve bir önceki Dokuz kesişim Modelindeki *boş küme* ya da *boş*

küme değil şeklindeki kesişim sonuçları kesişimlerin boyutları cinsinden değerlendiren bir modeldir.

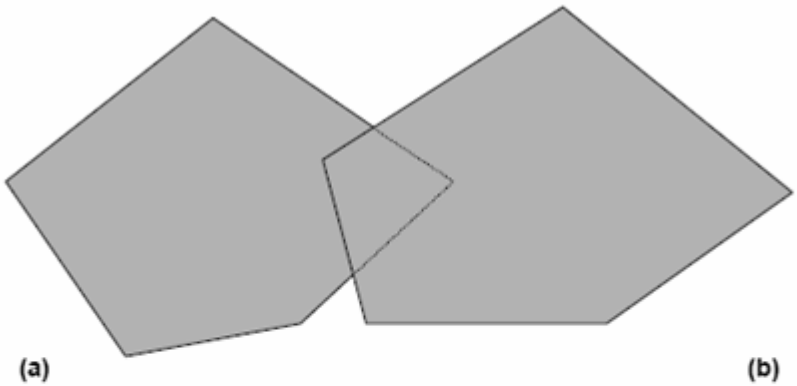
Bir a geometrisinde, *I(a)*, *B(a)* ve *E(a)* sırasıyla a'nın İç, Sınır ve Dışını temsil etsin. *I(a)*, *B(a)* ve *E(a)* 'nın herhangi ikisinin kesişmesi farklı boyutlarda geometriler kümesi olan, x kümesini 'i verir.

Örneğin, iki poligonun sınırlarının kesişimi bir nokta ve bir çizgiden oluşabilir. *dim(x)* fonksiyonu x'teki geometrilerin maksimum boyutunu (-1, 0, 1, ya da 2) döndürür, burada -1 sayısal değeri *dim(Ø)* ye karşılık gelecektir. Boyutsal olarak uzatılmış dokuz köşeli bir matrisin (DE-9IM) bu durumda Tablo 3'deki gibi olacaktır.

	İç	Sınır	Dış
İç	$dim(I(a)\cap I(b))$	$dim(I(a)\cap B(b))$	$dim(I(a)\cap E(b))$
Sınır	$dim(B(a)\cap I(b))$	$dim(B(a)\cap B(b))$	$dim(B(a)\cap E(b))$
Dış	$dim(E(a)\cap I(b))$	$dim(E(a)\cap B(b))$	$dim(E(a)\cap E(b))$

Tablo 3 Boyutsal olarak genişletilmiş 9 kesişim matrisi

Boyutsal olarak uzatılmış 9 kesişme matrisine bir örnek Şekil (2.12) 18'de verilmiştir.



	İç	Sınır	Dış
İç	2	1	2
Sınır	1	0	1
Dış	2	1	2

Şekil (2.12) 18 a ve b nesnelerinin 9 kesişim matrisi

Eğer iki geometri arasındaki uzamsal ilişki patern matrisi tarafından temsil edilen kabul edilebilir değerlerin birine karşılık geliyorsa, bu durumda sonuç DOĞRU (TRUE) çıkar. Eğer kesişim kümesi boş ise, bu durumda dim(Ø) fonksiyonunun değeri -1 olacaktır ve buda matriste YANLIŞ (FALSE) olarak temsil edilir.

Matris 9 adet dim(x) fonksiyonundan oluşur. Bu fonksiyonların olabilecek değerleride *p* {T, F, *, 0, 1, 2}'dir. Esasen bunların anlamı şöyledir:

$p = T \Rightarrow dim(x) \in \{0,1,2\}$

$p = F \Rightarrow dim(x) = -1$

$p = * \Rightarrow dim(x) \in \{-1,0,1,2\}$

$p = 0 \Rightarrow dim(x) = 0$

$p = 1 \Rightarrow dim(x) = 1$

$p = 2 \Rightarrow dim(x) = 2$

Patern matrisi dokuz karakter listesi şeklinde ifade edilebilir. Bir örnek için iki alanın örtüşen ilişkisini inceleyelim;

```
Char * overlapMatrix = "T*T***T**";
Geometri* a, b;
Boolean c = a-> Relate(b, overlapMatrix);
```

2.2.4 Dokuz-Kesişim modeline göre konumsal ilişki önermelerinin isimlendirmesi

Dokuz-kesişim metodundaki önermelere birçok geliştiricinin alıştığı ve yakın olduğu anlamlı isimler vermeliyiz. Bu amaçla 5 önerme isimlendirilmiştir; ayrık, değen, kesen, içinde ve örten (Disjoint, Touches, Crosses, Within and Overlaps).

Ayrık (Disjoint):

Topolojik olarak kapalı iki geometri a ve b

$a.ayrık(b) \Leftrightarrow a \cap b = Ø$

Dokuz-Kesişim Modeli türünden gösterilişi:

$a.ayrık(b) \Leftrightarrow (I(a) \cap I(b) = Ø) \wedge (I(a) \cap B(b) = Ø) \wedge (B(a) \cap I(b) = Ø) \wedge (B(a) \cap B(b) = Ø) \Leftrightarrow$ *a.ilişki(b,'FF*FF****)*

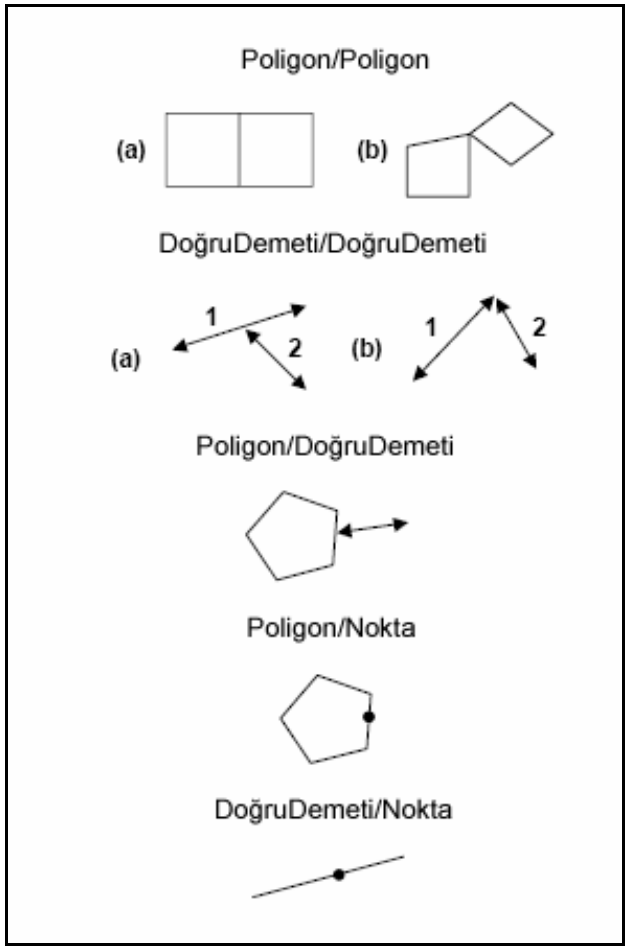
Değen (Touches):

Değen geometrik önermesi şu gruplarda geçerlidir; A-A, D-D, D-A, N-A ve N-D (A:Alan, D: Doğru, N:Nokta). Fakat N-N grubu için geçerli değildir. Şekil (2.13) 19'da bazı geçerli değen ilişkileri gösterilmiştir.

$a.değen(b) \Leftrightarrow (I(a) \cap I(b) = Ø) \wedge (a \cap b) \neq Ø$

Dokuz-Kesişim Modeli türünden gösterilişi:

*a.değen(b) ⇔ (I(a) ∩ I(b) = Ø) ^ (B(a) ∩ I(b) ≠ Ø) v (I(a) ^ B(b) ≠ Ø) v (B(a) ∩ B(b) ≠ Ø) ⇔ a.ilişki(b, 'FT*******') v a.ilişki (b, 'F**T*****') v a.ilişki(b, 'F***T****')*



Şekil (2.13) 19 Bazı değen ilişkileri.

Kesen (Crosses):

Şekil (2.14) 20'de örneklenen kesen ilişkisi N-D, N-A, D-D ve D-A nesne çiftleri için geçerlidir (A:Alan, D: Doğru, N:Nokta) ve şu şekilde ifade edilir:

a.Kesen(b) ⇔ (dim((I(a) ∩I(b)) < enBüyük(dim(I(a)), dim(I(b)))) ^ (a ∩ b≠ a) ^ (a∩ b≠ b)

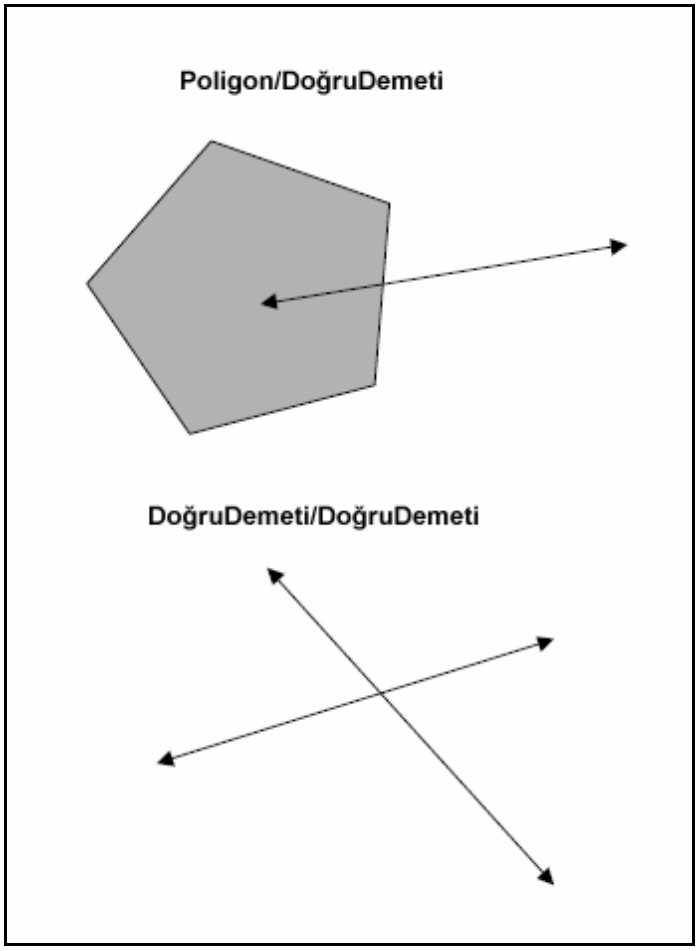
Dokuz-Kesişim Modeli türünden gösterilişi:

Durum; a∈N, b∈D veya Durum; a∈N, b∈A veya a∈D, b∈A:

*a.Kesen(b) ⇔ (I(a)∩I(b) ≠ Ø) ^ (I(a) ∩E(b)≠ Ø) ⇔ a.ilişki(b ,"T*T******)*

Durum; a∈D, b∈D:

*a.Kesen(b) ⇔ dim(I(a) ∩I(b)) = 0 ⇔ a.İlişki(b,"0********")*



Şekil (2.14) 20 Kesen İlişkisine bazı örnekler.

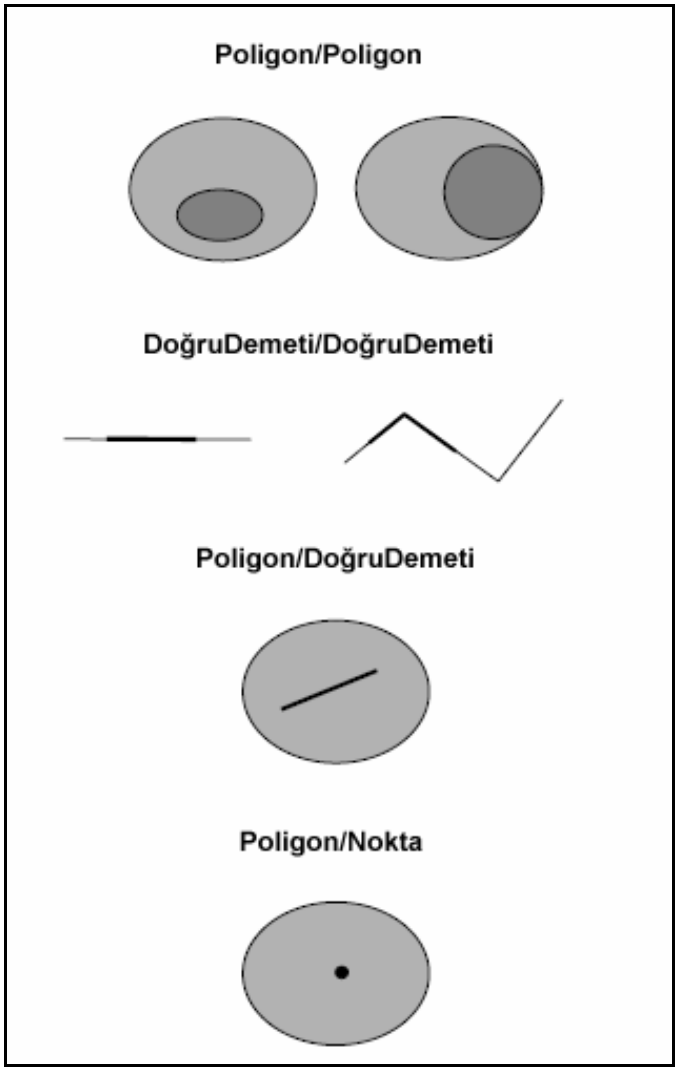
İçinde (Within):

Şekil (2.15) 21'de örneklenen içinde ilişkisi şu şekilde tanımlanır:

a.İçinde(b) ⇔ (a∩b = a) ^ (I(a) ∩I(b)≠ Ø)

Dokuz-Kesişim Modeli türünden gösterilişi:

*a.İçinde(b) ⇔ (I(a) ∩I(b) ≠ Ø) ^ (I(a) ∩E(b) = Ø) ^ (B(a) ∩E(b) = Ø)) ⇔ a.ilişki(b,"T*F**F***")*



Şekil (2.15) 21 İçinde ilişkisine bazı örnekler.

Örtüşen (Overlaps):

Örtüşen ilişkisi A-A, D-D ve N/N nesne çiftleri için geçerlidir (A:Alan, D: Doğru, N:Nokta). Şekil (2.16) 22'de örtüşen ilişkisine bazı örnekler verilmiştir.

Şu şekilde tanımlanır:

a.Örtüşen(b) $\Leftrightarrow$ (dim(I(a)) = dim(I(b)) = dim(I(a)$\cap$I(b))) ^ (a$\cap$b$\neq$a) ^ (a$\cap$b$\neq$b)

Dokuz-Kesişim Modeli türünden gösterilişi:

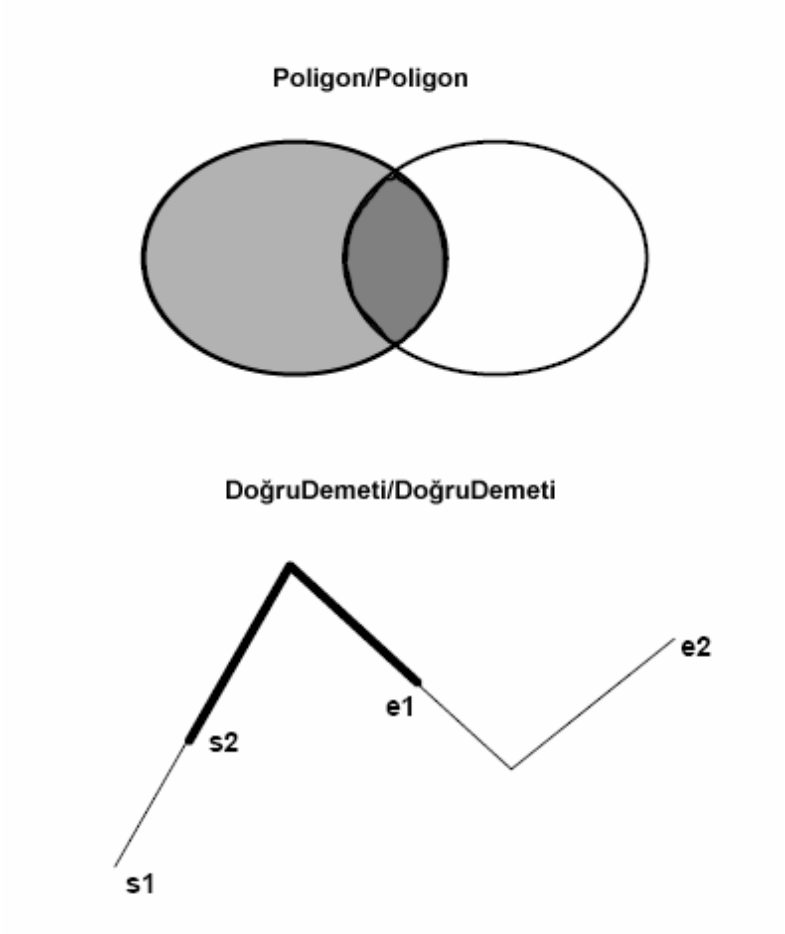
Durum; *a$\in$N, b$\in$N veya Durum; a$\in$A, b$\in$A:*

*a.Örtüşen(b)$\Leftrightarrow$ (I(a) $\cap$ I(b) $\neq$ Ø) ^ (I(a) $\cap$ E(b)$\neq$ Ø) ^ (E(a) $\cap$ I(b)$\neq$ Ø) $\Leftrightarrow$ a.ilişki(b,"T*T***T**)*

Durum; $a \in D,\ b \in D$:

$a.Örtüşen(b) \Leftrightarrow (\dim I(a) \cap I(b) = 1) \wedge (I(a) \cap E(b) \neq \emptyset) \wedge (E(a) \cap I(b) \neq \emptyset)$

$\Leftrightarrow$ *a.ilişki(b,"1*T***T**)*



Şekil (2.16) 22 Örtüşen ilişkisine bazı örnekler.

İçerir (Contains):

$a.İçerir(b) \Leftrightarrow b.İçinde(a)$

Kesişir (Intersects):

$a.Kesişir(b) \Leftrightarrow !a.Ayrık(b)$

Yukarıdaki operatörler temel alınarak OpenGIS tarafından Geometri sınıfına şu metotlar yüklemiştir [35] : **Equals, Disjoint, Intersects, Touches, Crosses, Contains, Overlaps, Relate**

Bölüm 3 - Konumsal Sınıf Kütüphanesi

Bu bölümde geliştirilen konumsal sınıf kütüphanesinin özellikler kümesi anlatılacak ve OpenGIS'in standartlarını (Simple Features Specification (SFS) [35]) nasıl sağladığına değinilecektir.

3.1 Geometrik Nesneler

Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL Revision 1.1 (SFS) 'de [35] bazı geometrik nesneler ve bunların tanımlamaları bulunmaktadır. Bu gerçekleştirilecek bir kütüphanenin bu tanımlamalara uyması gerekmektedir. Eğer uyulmayan bir tanımlama varsa bunun açıklanması gerekmektedir.

Geometri

SFS ye göre geometri nesnesi sadece kapalı kümeleri temsil eder. Yani farklı tanımlamaları olan kaynaklardan gelen nesneleri temsil edemez. Bu, pratik uygulamalara olanak sağlayan makul bir karardır.

Boş Geometri

SFS, her Geometri alt sınıfı nesnesinin boş olabileceğini açıkça belirtir. Bu bazen sınıf Geometrinin genel boş nesnesini kurmak için zorunludur. (Örneğin, eğer geri gelecek Geometrinin tam çeşidi bilinmiyorsa). SFS, genel boş Geometriyi temsil eden özel bir sınıf ya da nesne tanımlamamıştır.

Geometri Koleksiyonu

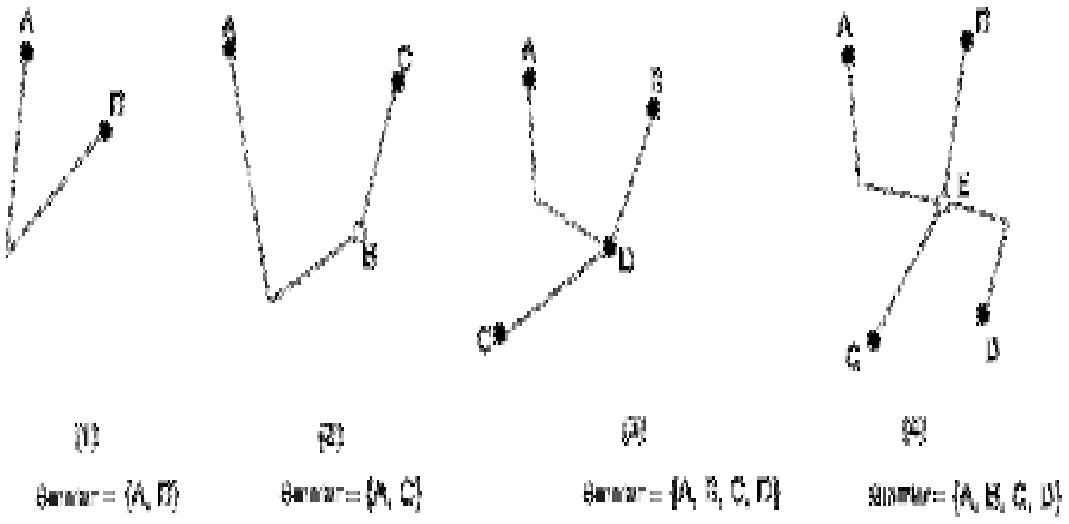
Bir heterojen geometri koleksiyonunun boyutu elemanlarının maksimum boyutudur.

Eğri

Boş olmayan eğri en az 2 noktaya sahip olmalıdır ve hiç bir ardışık 2 nokta eşit olmamalıdır.

Çoklu Eğri

SFS, çoklu eğrinin sınırlarını belirlemede "Mod-2"kuralının kullanımını açıkça belirtmiştir. Mod-2 kuralına göre eğer bir nokta çoklu eğrinin tek sayıda elemanlarının sınırları üzerindeyse, bu nokta çoklu eğrinin sınırları üzerindedir. Örneğin, Şekil (3.1) 23(3) de, B noktası SFS' ye göre sınırdadır, ancak nokta-küme topolojisine göre iç noktadır.

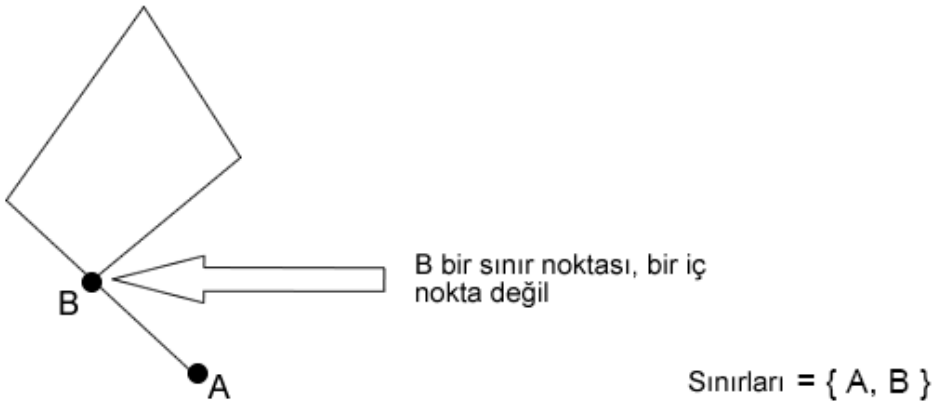


Şekil (3.1) 23 Mod-2 kuralının çoklu doğru katarlarındaki etkisi

Doğru Katarı

Kütüphanede OGC SFS 'de verilen Doğru Katarı tanımını kullanılmaktadır. Bu, diğer bazı uzaysal modellerden önemli şekilde ayrılır (örneğin; ESRI ArcSDE tarafından kullanılan). Doğru katarı basit olmayabilir. Noktalarda ya da doğru

parçalarında kendileriyle kesişebilir. Gerçekten eğrinin sınır noktaları Şekil (3.2) 24'deki gibi (örneğin bitiş noktaları) eğrinin içi ile kesişebilir. Böylece, teknik olarak topolojik olarak kapalı fakat SFS' ye göre kapalı olamayan bir eğri ile sonuçlanabilir. Bu durumda topolojik olarak kesişme noktası eğrinin sınırları üzerinde olmamalıdır. Fakat SFS tanımına göre nokta sınırın üzerinde düşünülür.



Şekil (3.2) 24 Sınır noktası iç noktası ile kesişen bir doğru katarı

Doğrusal Halka

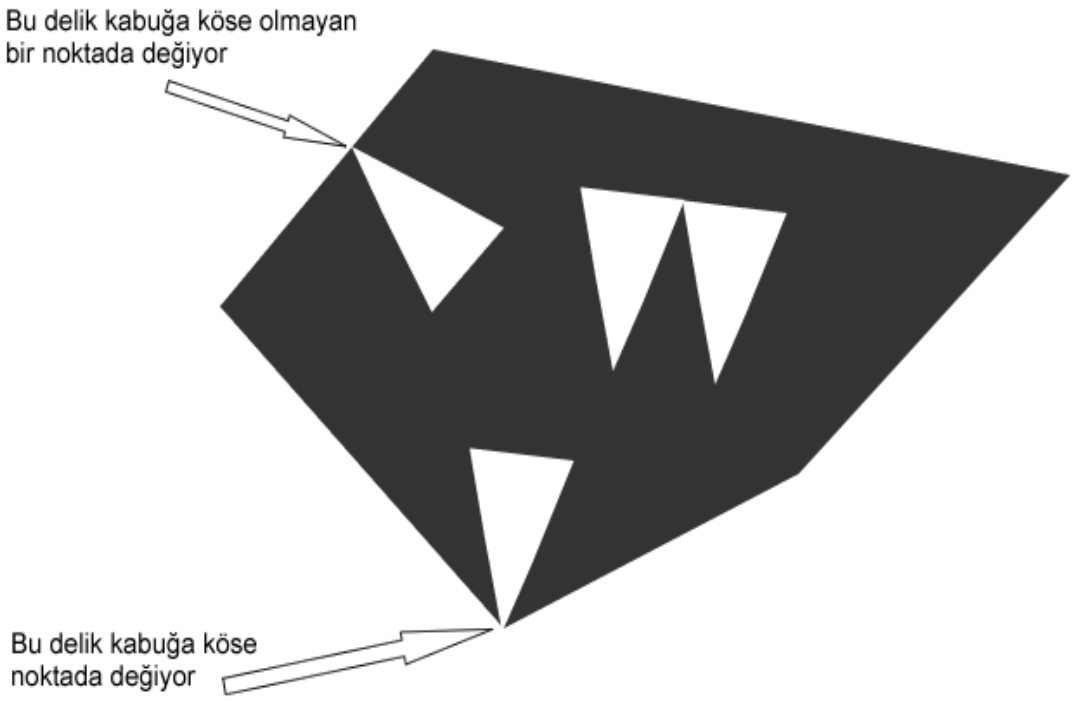
Doğrusal halkalar poligonlar için temel yapı taşlarıdır. Bir doğrusal halka en az 3 noktaya sahip olmalıdır. Doğrusal halkalar basit olmalıdır. Örneğin, bütün noktalar aynı doğru üzerinde olmamalı ve çember kendisiyle kesişmemelidir. SFS doğrusal halka yönlendirme ihtiyacını belirtmemiştir. Bu geometrik kütüphanede, doğrusal halkanın ya saat yönünde ya da saat yönünün tersine yönlendirilmesine olanak tanınmaktadır (burada yönlendirmeden kasıt noktaların koordinat dizisinde sıralanma yönüdür).

Poligon

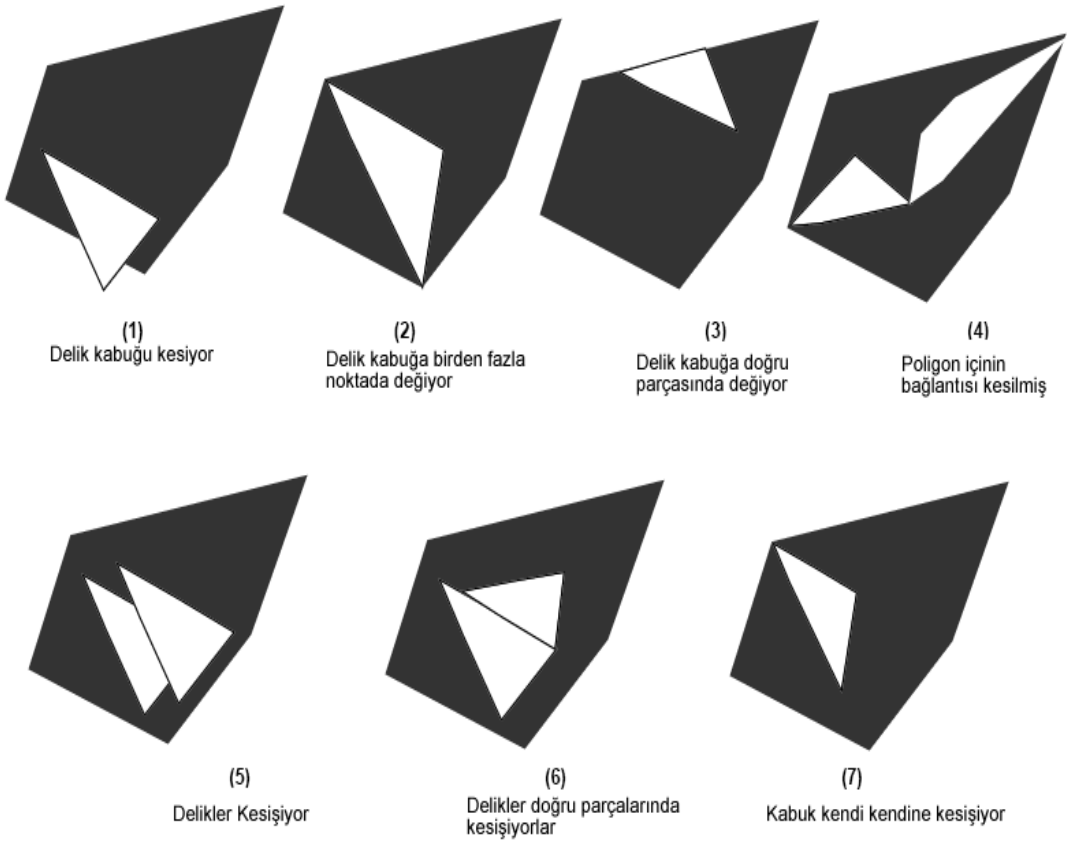
Poligonun kabuğu ve delikleri doğrusal halkadır. Poligonun SFS tanımı aşağıdaki anlamları içerir:

- Kabuk ve delikler kendileriyle kesişemez (bu doğrusal halka olma gerçeği tarafından belirtilmiştir).
- Delikler sadece kabuğa ya da tek bir noktadaki bir diğer deliğe değebilir. Bu, delikler birden fazla noktada ya da bir doğru parçasında birbirleriyle kesişemez demektir.
- Poligonun iç kısımları birbiriyle bağlı olmalıdır (Bu bir önceki ifadeyle belirtilmiştir).
- Bir deliğin kabuğa değdiği noktanın köşe olması gerekliliği yoktur. (bu duruma bir örnek şekil 29'da verilmiştir.

Poligonun SFS tanımının sıkça kullanılan bazı diğer uzaysal modellerden farklılaştığını unutmamak gerekir. Örneğin, ESRI ArcDE uzaysal modeli kabuğun köşelerde kendisiyle kesişmesine izin verirken, deliklerin kabuğa değmesine izin vermez. Şekil (3.3) 25'de delikli ve geçerli poligon örnekleri gösterilmektedir. Şekil (3.4) 26'da ise poligon olma ölçütlerine uymayan geometrik nesneler örneklenmiştir.



Şekil (3.3) 25 Delik içeren bir poligon örneği



Şekil (3.4) 26 Poligon olarak sunulamayacak örnekler

Poligonun kabuğu ve delikleri doğrusal halka olduğu için koordinatlarının bir yönde dizilmelerinin bir gereklilik yoktur. Hem saat yönünde hem de saat yönünün tersinde dizilebilirler.

Çoklu Poligon

Çoklu poligondaki eleman poligonlar sadece sonlu sayıda noktaya değebilir(Örneğin, doğru parçasında değmemelidirler). Elemanların iç kısımları kesişmemelidirler.

3.2 Yardımcı Sınıflar

Koordinat (Coordinate):

Koordinatları tutan basit bir sınıftır. Sınıf kütüphanesinde her cisim kendi x ve y değerlerini tutmak yerine Coordinate sınıfından yararlanırlar.

Koordinatlar iki boyutlu noktalardır. Bu noktalara opsiyonel bir de z koordinatı eklenebilir. Bu tez kapsamındaki Geometri Sınıfı kütüphanesinde z koordinatı bulunmamaktadır.

Sıralı Koordinat (Coordinate Sequence):

Bir koordinatlar listesinin bir Geometri içindeki iç gösterimidir. Esasen bu sınıf, koordinatlar dizisi oluşturmak için kullanılır.

Kaplayan Dörtgen (Envelope):

Bu sınıf geometrik nesnelerin iki boyutlu uzayda dikdörtgensel sınırlarını tanımlar. Bu dikdörtgen Şekil (3.5) 27'deki gibi o nesneyi içine tamamen alabilecek en küçük dikdörtgendir. Özellikle dizinleme işlemlerinde Envelope sınıfı büyük önem taşır.

Şekil (3.5) 27 En küçük sınırlayıcı dikdörtgen

Kesişim Matrisi (Intersection Matrix):

Boyutsal olarak genişletilmiş 9 kesişim matrisinin gerçekleştirimidir. Bölüm 2'de anlatılmış olan 9-kesişim matrisi kuramını hayata geçirir.

Evliya-Well-Known Text:

Geometrik nesnelerin bir veri tabanında tutulabilmeleri için yine OpenGIS tarafından tanımlanmış Well known Text adlı bir standart bulunmaktadır [33] . Geometrik bir nesnelerin şekil bilgileri sayılabilir sınırsız olabileceğinden (ör. Ankara il sınırları) bunları veri tabanında Şekil (3.6) 28'deki gibi tek bir alanda tutabilmek için Well Known Text isimli bir yalın yazı nesne tanımlama formatı kullanılmaktadır.

GID	Xmin	Ymin	Xmax	Ymax	WKT_Geometry
1	7	4	40	88	POLYGON((10 10, 10 20,…))
2	10	10	20	20	MULTIPOINT(10 10, 20 20)

Şekil (3.6) 28 Well Known Binary formatının veri tabanında kullanımı

WKT Geometrik sınıf kütüphanesindeki her sınıfı ifade edebilmektedir. WKT formatına birkaç örnek vermek gerekirse:

POINT(10 10) – Bir Nokta

LINESTRING(10 10, 20 20, 30 40) – Üç noktalı bir doğru katarı

POLYGON((10 10, 10 20, 20 20,20 15, 10 10)) - Bir dış birde iç yüzüğü olan bir poligon.

MULTIPOINT(10 10, 20 20) – İki noktalı bir çoklu nokta.

MULTILINESTRING((10 10, 20 20), (15 15, 30 15)) – İki doğru katarı olan bir çoklu doğru katarı

MULTIPOLYGON(((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 70, 80 60, 60 60))) – İki poligonlu bir çoklu poligon

GEOMETRYCOLLECTION(POINT (10 10),POINT (30 30), LINESTRING (15 15, 20 20)) – İki nokta ve bir doğru katarı olan bir geometri koleksiyonu

Verilen bir geometriden nesneleri yaratabilmek için WKT çözümleyicisine ihtiyaç duyulmaktadır. Bu çözümleyiciyi geliştirebilmek içinde WKT formatını *Bechus-Naur form'a* (BNF) dönüştürülmüştür.

<Geometry Tagged Text> :=

| <Point Tagged Text>
| <LineString Tagged Text>
| <Polygon Tagged Text>
| <MultiPoint Tagged Text>
| <MultiLineString Tagged Text>
| <MultiPolygon Tagged Text>

<Point Tagged Text> :=
POINT <Point Text>

<LineString Tagged Text> :=
LINESTRING <LineString Text>

<Polygon Tagged Text> :=
POLYGON <Polygon Text>

<MultiPoint Tagged Text> :=
MULTIPOINT <Multipoint Text>

<MultiLineString Tagged Text> :=
MULTILINESTRING <MultiLineString Text>

<MultiPolygon Tagged Text> :=
MULTIPOLYGON <MultiPolygon Text>

<Point Text> := EMPTY
| <Point>
| Z <PointZ>
| M <PointM>
| ZM <PointZM>

<Point> := <x> <y>
<x> := double precision literal
<y> := double precision literal

<PointZ> := <x> <y> <z>
<x> := double precision literal
<y> := double precision literal
<z> := double precision literal

<PointM> := <x> <y> <m>
<x> := double precision literal
<y> := double precision literal
<m> := double precision literal

<PointZM> := <x> <y> <z> <m>
<x> := double precision literal
<y> := double precision literal
<z> := double precision literal
<m> := double precision literal

<LineString Text> := EMPTY
| (<Point Text > {, <Point Text> }*)
| Z (<PointZ Text > {, <PointZ Text> }*)
| M (<PointM Text > {, <PointM Text> }*)
| ZM (<PointZM Text > {, <PointZM Text> }*)

<Polygon Text> := EMPTY
| (<LineString Text > {,< LineString Text > }*)

<Multipoint Text> := EMPTY
| (<Point Text > {, <Point Text > }*)

<MultiLineString Text> := EMPTY
| (<LineString Text > {,< LineString Text>}*)

<MultiPolygon Text> := EMPTY
| (< Polygon Text > {, < Polygon Text > }*)

BNF gösterimi sayesinde WKT formatındaki verileri gerçek nesnelere çeviren Evliya-WKT isimli sınıf gerçekleştirilmiştir.

3.2 Ana Sınıflar:

Geometri kütüphanesindeki SFS' de bulunması şart olan ve kütüphanenin omurgasını oluşturan sınıflar, temel işlemleri ve metotları bu bölümde anlatılmaktadır.

3.2.1 Geometri Sınıfı

Geometri sınıf kütüphanesinin kök sınıfıdır. Geometri sınıfı esasen soyut bir sınıf olup gerçekte doğrudan tanımlanamaz. Tanımlanabilecek alt sınıfları 0, 1 ya da 2 boyutlu ve R^2'de tanımlı nesneler olmalıdır.

Geometri sınıfının temel metotları şunlardır:

Boyut (Dimension): Geometri nesnesinin miras aldığı boyutu tutar. Bu sınıf kütüphanesinde nesneler 2 boyutlu uzay ile sınırlandırılmışlardır.

Geometri Tipi (GeometryType): Alt geometrinin tipini döndürür. Burada her geometrinin kendini tanımlayabilmesi önemlidir.

Örten Kutu (Envelope): Bu metot kendisini sınırlayan en küçük kapalı kutuyu bulur.

Yazı Olarak (AsText): Kendisini tanımlayan yazı gösterimini geri döndürür.

Boş Mu (IsEmpty): Eğer geometri boş nokta kümesi ise doğru değerini döndürür.

Basit Mi (IsSimple): Geometrinin basit bir geometri olup olmadığını döndürür. (Basit geometriler 2. bölümde anlatılmıştır).

Sınır (Boundary): Kendisinin sınırlarını döndürür.

Diğer bir geometrik nesne ile konumsal ilişkileri test eden metotlar:

Eşitlik (Equals): Kendisinin verilen bir diğer geometri ile konumsal olarak eşit olup olmadığını sınar.

Ayrıklık (Disjoint): Kendisinin verilen bir diğer geometri ile ayrık olup olmadığını sınar.

Kesişme (Intersects): Verilen bir diğer geometri ile kendisinin kesişip kesişmediğini sınar.

Değme (Touches): Verilen bir diğer geometri ile sınırlarının değip değmediğini sınar.

Çaprazlama (Crosses): Verilen başka bir geometri ile kendisinin çaprazlanıp çaprazlanmadığını sınar.

İçinde (Within): Kendisinin verilen diğer geometrinin içinde olup olmadığını sınar.

İçerir (Contains): Kendisinin verilen başka bir geometriyi kapsayıp kapsamadığını sınar.

Örter (Overlaps): Kendisinin verilen başka bir geometri ile üst üste gelip gelmediğini sınar.

Konumsal Analizleri Destekleyen Metotlar:

Uzaklık (Distance): Verilen bir nesne ile arasındaki en yakın iki nokta arasındaki mesafeyi hesaplar.

Dış Bükey Kabuk (ConvexHull): Kendisinin dış bükey omurgasını temsil edecek bir başka geometri nesnesi bulup döndürür.

Kesişim (Intersection): Başka bir geometri ile kesişimleri sonucu ortaya çıkan noktalar kümesini bulur.

Birleşim (Union): Başka bir geometri ile birleşimi sonucu oluşan noktalar kümesini döndürür.

Fark (Difference): Diğer bir geometriden farkını bulup bunu noktalar kümesi olarak döndürür.

3.2.2 Geometri Koleksiyonu Sınıfı

Geometri koleksiyonu esasen içinde bir ya da daha fazla geometrinin bulunduğu bir geometridir.

Geometri Koleksiyonu sınıfının bazı kendine has metotları:

Geometri Sayısı (NumGeometries): İçerisinde kaç adet geometri barındırdığını belirtir.

N. Geometri (GeometriN): N. Geometriyi geri döndürür.

3.2.3 Nokta Sınıfı

Nokta, koordinat boşluğunda tek bir yeri gösteren sıfır boyutlu bir geometridir. Noktanın sınırları boş kümedir.

Nokta sınıfının kendine has metotları:

X: Kendisinin x koordinatını döndürür.

Y: Kendisinin y koordinatını döndürür.

3.2.4 Çoklu Nokta Sınıfı

Çoklu nokta, sıfır boyutlu geometri koleksiyonudur. Elemanları sadece noktalar olabilir. Eğer nokta koleksiyonundaki hiçbir iki nokta aynı koordinat noktalarına sahip değillerse bu nesne basit bir çoklu noktadır denir.

3.2.5 Eğri Sınıfı

Eğri genellikle sıralı noktalardan oluşan tek boyutlu bir nesnedir. Eğer eğri bir noktadan iki kez geçmiyorsa basittir.

Topolojik olarak

$c.IsSimple \Leftrightarrow (\forall\, x1,x2 \in (a, b]\ x1 \neq x2 \Rightarrow f(x1) \neq f(x2)) \cap (\forall\, x1,x2 \in [a, b)\ x1 \neq x2 \Rightarrow f(x1) \neq f(x2))$

Eğer eğrinin başlangıç noktası bitiş noktasına eşitse eğri kapalıdır.

Bir eğri basit ve kapalı ise o eğri bir halkadır.

Eğrinin kendine has metotları:

Uzunluk (Length): Eğrinin uzunluğunu hesaplayan metot.

Başlangıç Noktası (StartPoint): Eğrinin başlangıç noktası.

Bitiş Noktası (EndPoint): Eğrinin bitiş noktası.

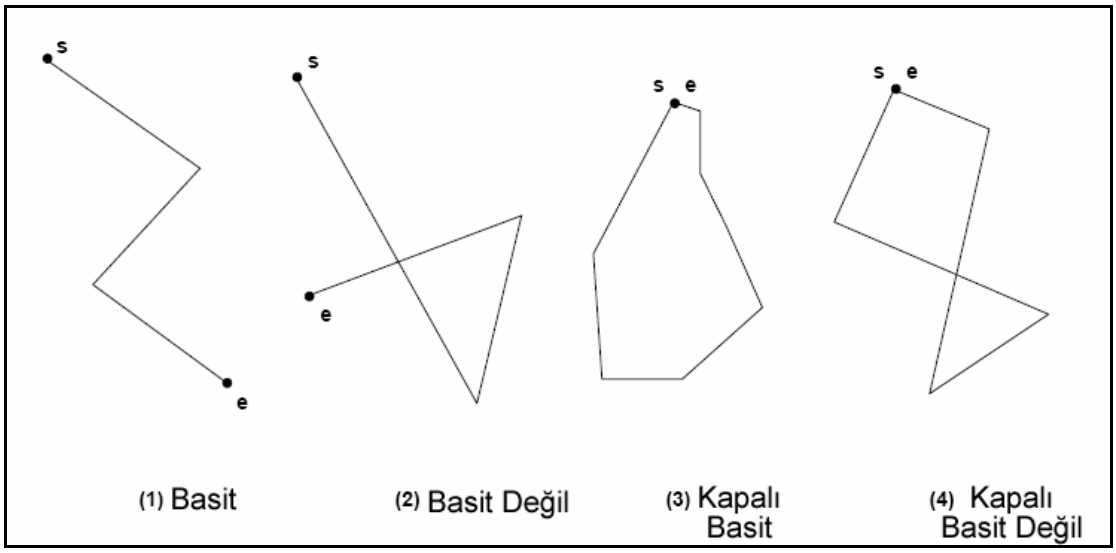
Kapalı Mı (IsClosed): Eğrinin kapalılığını sınayan metot.

Halka Mı (IsRing): Eğrinin bir halka olup olmadığını sınayan metot.

3.2.6 Doğru Demeti, Doğru ve Doğrusal Halka Sınıfları

Bir Doğru Demeti (Linestring) bir birini takip eden noktaların birleşmesi sonucu oluşan bir eğridir (Curve). Her takip eden nokta esasen bir çizgi oluşturmaktadır.

- Bir doğru sadece iki noktadan oluşan bir doğru demetidir (LineString).
- Bir doğrusal halka (LinearRing) kapalı ve basit bir doğru demetidir.



Şekil (3.7) 29 Doğru demeti örnekleri

Şekil (3.7) 29'de 4 adet Doğru demeti görülmektedir. Bunların gerçekten bir Doğrusal halka olup olmadığına karar vermek gerekmektedir.

1.Şekil basit fakat kapalı değildir, dolayısıyla doğrusal halka değildir.

2.Şekil kapalı ve basit değildir, dolayısıyla doğrusal halka değildir.

3.Şekil hem kapalı hem de basit bir doğru demetidir, dolayısıyla bir doğrusal halkadır.

4.Şekil ise kapalı fakat basit bir doğru demeti değildir, dolayısıyla doğrusal halka değildir.

Buradan anlaşılacağı gibi Doğrusal Halka (LinearRing) sınıfı yaratılırken kapalı doğru demeti olup olmadığı, aynı zamanda da basit bir doğru demeti olup olmadığı kontrol edilmektedir.

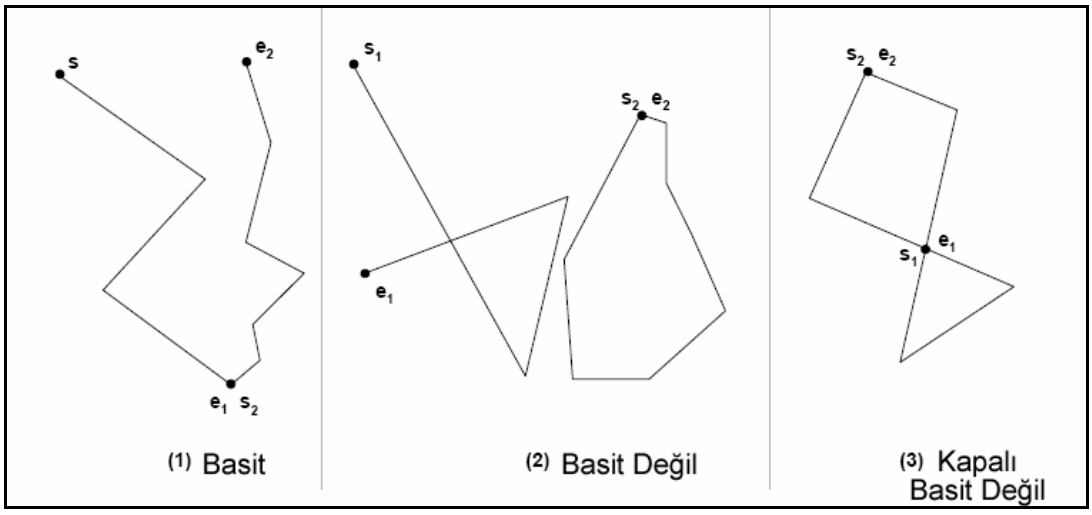
Sınıfın kendine has metotları:

Nokta Sayısı (NumPoints): Doğru demetinin içindeki nokta sayısını geri döndürür.

N. Nokta (PointN): N'inci noktayı geri döndürür.

3.2.7 Çoklu Doğru Demeti Sınıfı

Çoklu doğru demeti, elemanları çoklu doğrular olan çoklu eğriler kümesidir. Şekil (3.8) 30'de çoklu doğru demetine bazı örnekler verilmiştir.



Şekil (3.8) 30 Çoklu doğru demeti örnekleri

3.2.8 Poligon Sınıfı

Poligon bir dışsal, sıfır ya da daha çok içsel sınırlamaları olan düzlemsel bir alandır. Her içsel sınır poligon içinde bir delik tanımlar. Şekil (3.9) 31'de bazı geçerli, Şekil (3.10) 32'de ise bazı geçersiz poligon örnekleri verilmiştir.

Geçerli bir poligon tanımlamanın bazı kuralları vardır.

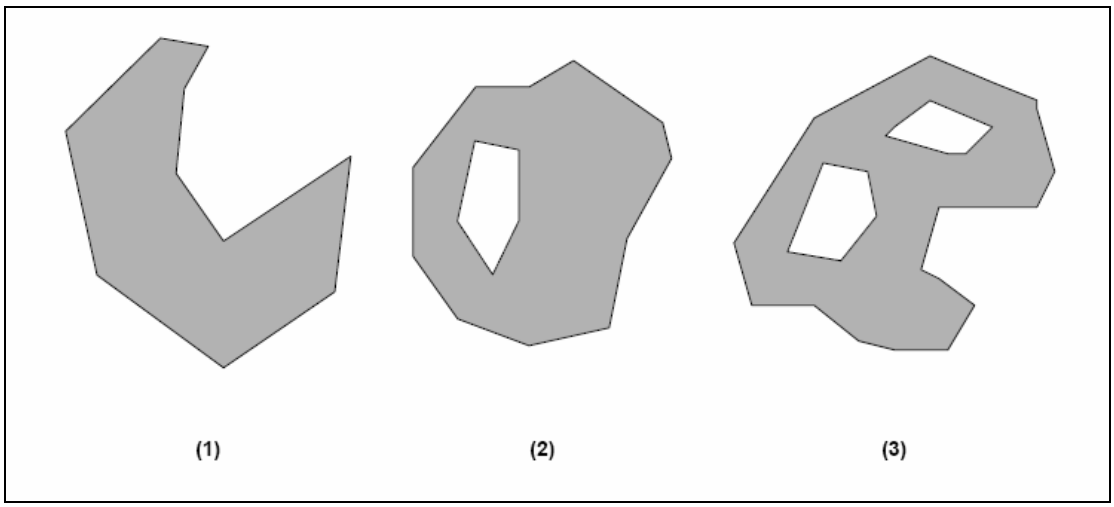
1. Poligonlar topolojik olarak kapalıdırlar.
2. Poligonun içsel ve dışsal sınır çizgileri doğrusal halka kümesinden oluşur.
3. Sınırdaki herhangi iki doğrusal halka çaprazlayamazlar (burada çaprazlamaktan kasıt kesen fakat değmeyendir). Sınırdaki halkalar bir noktada kesişebilir fakat sadece tanjant olarak. Yani;

$\forall P \in Poligon, \forall c1,c2 \in P.Sınır(), c1 \neq c2 , \forall p, q \in Nokta, p, q \in c1, p \neq q , [p \in c2 \Rightarrow q \notin c2]$

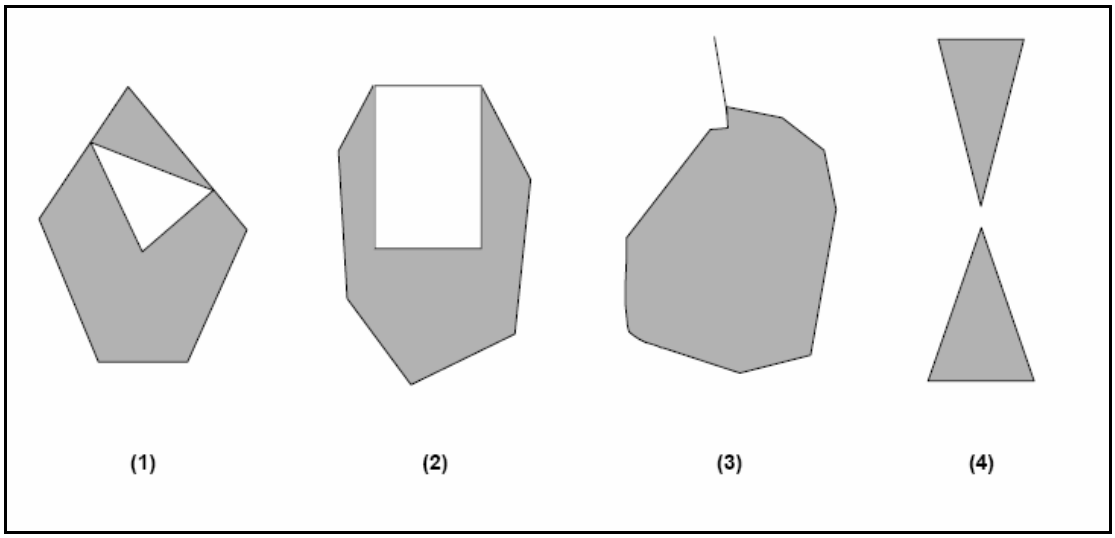
4. Bir poligonun kendini kesen çizgisi, dışa ya da içe doğru sivriliği olamaz. Yani;

$\forall$ P $\in$ Poligon, P = Kapalılık(İçsel(P))

5. Poligonun içi birbirine bağlı nokta kümesinden oluşur.
6. Bir ya da daha çok iç delik birbirleriyle bağlantılı olmaz.



Şekil (3.9) 31 Geçerli poligon örnekleri



Şekil (3.10) 32 Geçersiz poligon örnekleri

Şekil (3.9) 31 'de geçerli poligon örnekleri görülmektedir. Şekil (3.10) 32'de ise yukarıdaki kurallara uymayan poligon örnekleri gösterilmektedir. Şekil (3.10) 32'deki 1. ve 4. şekiller iki ayrı poligon olarak geçerli fakat tekbir poligon olarak geçersizdir.

Poligon sınıfının tanımlanan bazı kendine has metotları:

Dış Halka (ExteriorRing): Poligonun dışsal doğrusal yüzüğünü döndürür (LinearRing).

Delik Sayısı (NumInteriorRing): Poligonun içsel doğrusal halkalarının (delik) sayısını döndürür (LinearRing).

N. Delik (InteriorRingN): N'inci içsel yüzüğünü döndürür. Bu halka doğru demeti türündedir.

3.2.9 Çoklu Poligon Sınıfı

Çoklu Poligon elemanları poligon olan çoklu düzlemdir. Çoklu Poligon sınıfının da kendine özel kuralları vardır;

1. Çoklu poligondaki herhangi iki poligon birbirlerini kesmemelidir. $\forall M \in$ *Çoklu Poligon,* $\forall$ *Pi, Pj* $\in$ *M.Geometriler(),* $i \neq j$, *İçsel(Pi)* $\cap$ *İçsel(Pj)* $= \Theta$

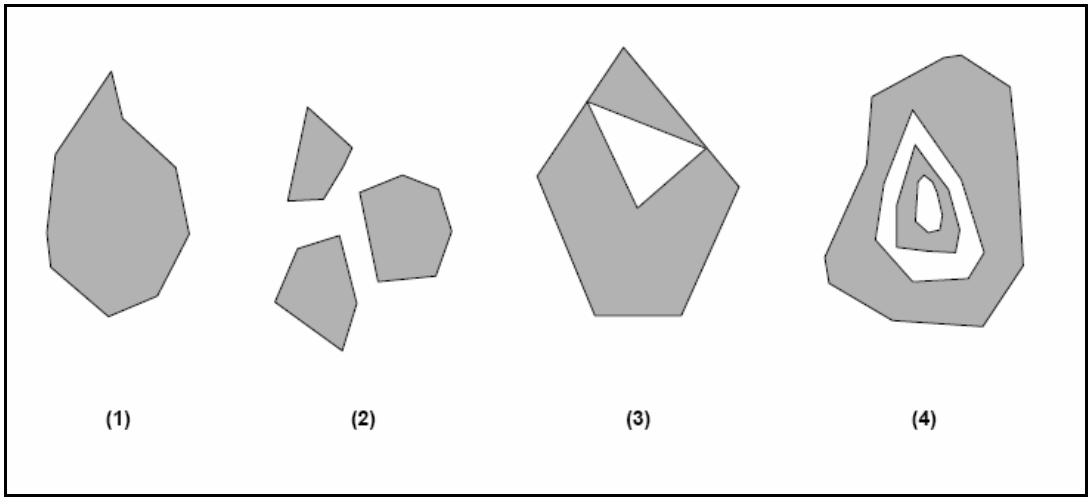
2. Çoklu poligonuna ait olan herhangi iki poligonun sınırları birbirlerini çaprazlayamaz, ayrıca sınır çizgileri sonlu sayıda noktada değebilirler. Yani; $\forall M \in$ *Çoklu Poligon,* $\forall$ *Pi, Pj* $\in$ *M.Geometriler(),* $\forall$ *ci* $\in$ *Pi.Sınırları(), cj* $\in$ *Pj.Sınırları(), ci* $\cap$ *cj = {p1,.....,pk | pi* $\in$ *nokta , 1<=i<=k}*

3. Bir çoklu poligon topolojik olarak kapalıdır.

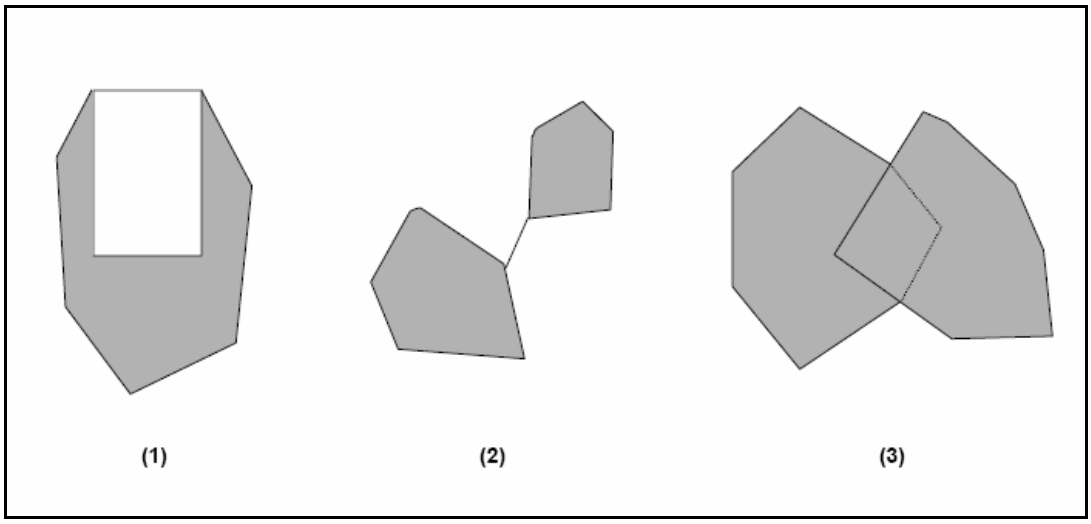
4. Çoklu poligonun kendini kesen çizgisi, dışa ya da içe doğru sivriliği olmaz. Çoklu poligon normal kapalı noktalar kümesidir. $\forall M \in$ *Çoklu Poligon, M = Kapalılık(İçsel(M));*

5. Çoklu poligonun içindeki bağlı bileşen sayısı, çoklu poligondaki poligon sayısına eşittir.

Bu kurallara uyan çoklu poligonlar örnekleri Şekil (3.11) 33'de verilmiştir.



Şekil (3.11) 33 Geçerli çoklu poligon örnekleri gösterilmiştir.



Şekil (3.12) 34 Geçersiz çoklu poligon örnekleri

Şekil (3.12) 34'de yukarıda verilen çoklu poligon maddelerine uymayan örnekler verilmiştir. Çoklu poligon sınıfının nesneleri bu kurallara uymak zorundadır.

Bölüm 4 - Algoritmalar

Bu geometri sınıf kütüphanesi yazılırken bazı önemli geometrik algoritmalardan yararlanılmıştır. Esasen bu algoritmalar geometrik sınıf kütüphanesindeki sınıfların birbirleri ile olan ilişkilerini işleyebilmesinde önemli rol oynamaktadır.

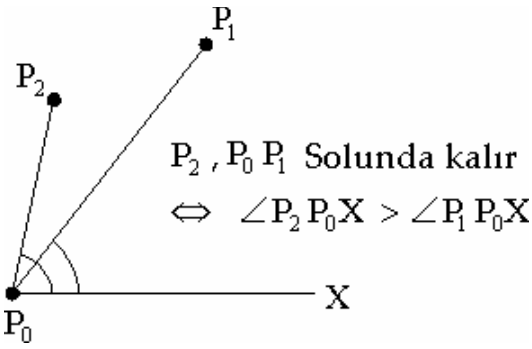
4.1 Dış Bükey Kabuk (Convex Hull)

Dış bükey kabuk uzayda bir şekli içine alan en küçük dış bükey çevre çizgisidir.

"Graham Scan" Algoritması:

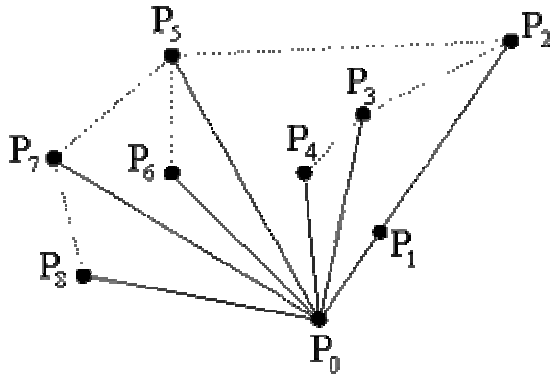
Graham Scan algoritması [23] Literatür'de ilk hesaplamalı grafik algoritması olarak göze çarpmaktadır. Nokta sayısı büyüdükçe (n = kümedeki nokta sayısı), algoritma asimptotik O(nlogn) karmaşıklık ile sınırlandırılabilir. Graham Scan Algoritmasında O(nlogn) zaman noktaları açısal sıraya sokarak geçmektedir.

S = {P} sonlu sayıda noktalar kümesi olsun. Algoritma S'in içinden dış bükey kabuktan bir nokta alarak başlar. Bu nokta cismin en sağ alt köşesinden seçilmektedir. Seçilen bu noktaya P_0 denilmektedir. Daha sonra algoritma Şekil (4.1) 35 'deki gibi diğer noktaları x ekseni ile olan açılarına göre saat yönü tersi seçmektedir. Eğer iki nokta aynı açıya sahip ise P_0'a en yakın olanı göz ardı edilmektedir.



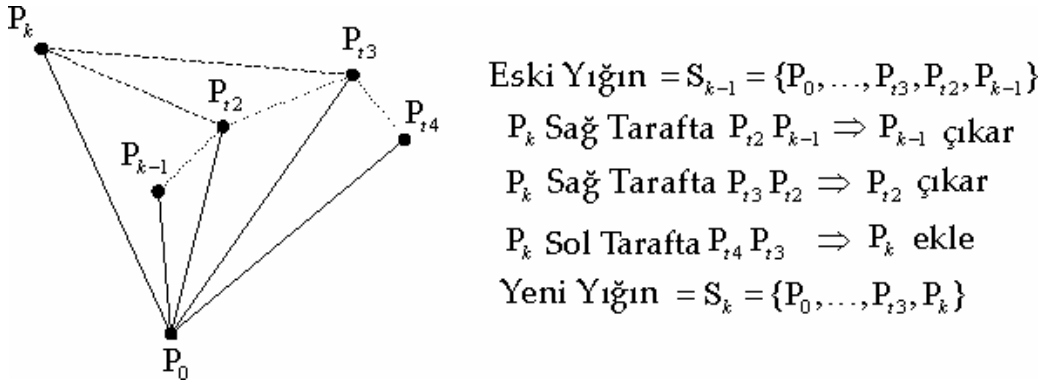
Şekil (4.1) 35 Graham Scan Algoritması ilk adım

Noktaları saat yönü tersi açısal olarak sıraya dizdikten sonra küme Şekil (4.2) 36'da gösterildiği gibi şuna benzemektedir; S = { P_0 , P_1 , P_2 , …, P_{n-1} }. Bu sıralama P_0 'ın merkezinde olduğu bir fana benzemektedir.



Şekil (4.2) 36 Graham Scan Algoritması ikinci adım

Daha sonra döngü ile S kümesindeki noktalar dış bükey kabuk testine tutulur. İlk iki noktanın oluşturdukları doğru ile üçüncü noktanın konumu bu testin sonucunu verir. Eğer üçüncü nokta doğrunun sağında kalıyorsa bu nokta dış bükey kabuğun içinde olamaz demektir ve sıralı noktaları tuttuğumuz yığın yapısından atılır. Eğer bu nokta doğrunun sol tarafında ise doğrunun sıralamaya göre son noktası yığın yapısından atılır.



Şekil (4.3) 37 Graham Scan Algoritması son adım

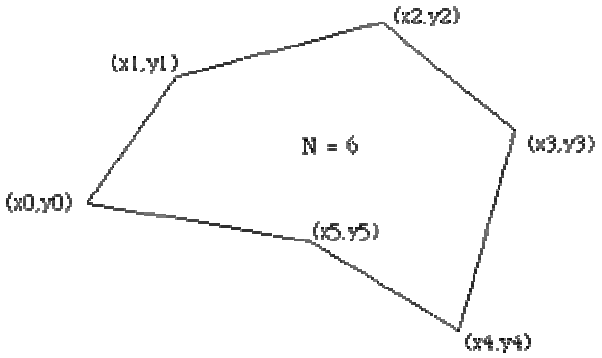
Sonunda Şekil (4.3) 37'deki gibi yığın yapısında sıralı olarak nesnenin dış bükey kabuk noktaları ortaya çıkar.

Bu algoritma kütüphanedeki poligon ve doğru katarı sınıflarında kullanılmıştır.

4.2 Bir Poligonun Alanı

Bir poligonun alanı CBS'lerde geometrik olarak poligon olan arazi, il, ilçe, şehir gibi birçok coğrafi nesnenin alan hesabında kullanılmaktadır. Poligonun alanı sınırlarının kapattığı bölgenin alanıdır.

Şekil (4.4) 38'deki gibi bir poligonun alanını bulma problemi karmaşık gibi görünse de esasen formül basittir. Varsayalım ki poligon *N* adet köşenin oluşturduğu doğru parçalarından meydana gelmektedir (x_i, y_i), i 0 dan N-1'e kadar gider. Son köşenin (x_n, y_n) ilk köşe ile aynı olduğu varsayılır [31] .



Şekil (4.4) 38 6 Köşeli poligon

Alan şöyle hesaplanır:

$$A = \frac{1}{2}\sum_{i=0}^{N-1}(x_i y_{i+1} - x_{i+1} y_i)$$

Delikli poligonlar için, deliklerin sıralamasının ters yapıldığı hesaba katılırsa bu algoritma hala geçerli olmaktadır. Tabi sonucun mutlak değeri alınmalıdır. Deliklerin alanı ters işaretli olacaktır.

Mutlak değeri alınmadan sonuca bakıldığında ise ilginç bir sonuç ortaya çıkmaktadır. Eğer poligon saat yönünde sıralanmışsa alan negatif, saat yönü tersi sıralanmışsa saat pozitif çıkmaktadır.

Bu algoritma kütüphanede poligon sınıfında kullanılmıştır.

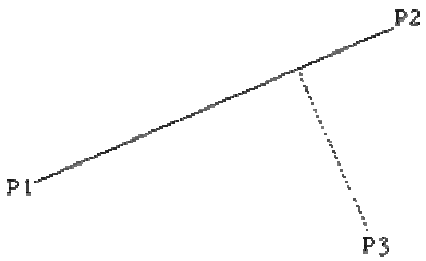
4.3 Bir nokta ve bir doğru arasındaki uzaklık

Bir nokta ve bir doğru arasındaki uzaklık hesabı diğer CBS kısımları tarafından sıkça kullanılacak bir işlemdir. Özellikle iki nesnenin uzaklık hesabında kullanılmaktadır. Burada bir nokta nesnesi ile bir doğru nesnesinin en yakın uzaklığı hesaplanmaktadır.

Şekil (4.5) 39'deki gibi P1-P2 doğrusu ile P3 noktası arasındaki en yakın mesafe P3 noktasının P1-P2 doğrusunu dik olarak kestiği doğru parçasının uzunluğudur.

$P_1(x_1, y_1)$ ve $P_2(x_2, y_2)$ noktalarının oluşturdukları doğrunun formülü

$\mathbf{P} = P_1 + u\,(P_2 - P_1)$



Şekil (4.5) 39 P1-P2 doğrusu ve P3 noktası

$P_3(x_3, y_3)$ noktasının $P_2 - P_1$ doğrusuna en kısa doğrusu $P_2 - P_1$ doğrusuna 90 derecelik açı yaptığı doğrudur. Oda nokta çarpımı ile hesaplanır.

$(P_3 - \mathbf{P})$ dot $(P_2 - P_1) = 0$

Verilen eşitliği açığımızda

$$u = \frac{(x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1)}{\|p_2 - p_1\|^2}$$

Eşitliği ortaya çıkar. Buradan kesişim noktaları ortaya çıkar

$x = x_1 + u\,(x_2 - x_1)$

$y = y_1 + u\,(y_2 - y_1)$

$P_2 - P_1$ doğrusu ile **P3** doğrusu arasındaki uzaklık P_3 ile (x,y) noktaları arasındaki uzaklığa eşittir.

Bu algoritma kütüphanedeki nokta ve doğru sınıflarında kullanılmıştır.

4.4 İki Doğrunun Kesişimi

İki doğrunun kesişim CBS'lerde nesneler arası ilişkileri tanımlamada ve bölüm 4.5'de anlatılan "Nokta Poligonun İçinde mi?" algoritmasını hayata geçirmede büyük önem taşımaktadır. Burada önemli olan iki doğru nesnesinin kesişip kesişmediğinin hesaplanması ve kesişiyorlarsa kesişim noktalarının bulunmasıdır.

Geometrik olarak iki doğru birbirine paralel değilse kesişir. Ancak CBS'lerde iki doğrunun değil iki doğru parçasının kesişimi önemlidir. İki doğrunun kesişiminin genel ifadesi

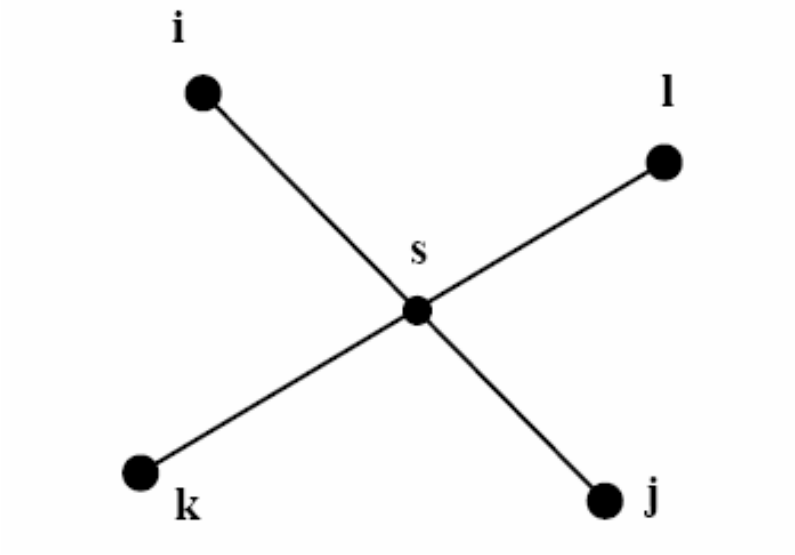
Birinci doğru Ax + By + C = 0

İkinci doğru Ex + Fy+ G = 0

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} (GB - FG)/(FA - EB) \\ (CE - AG)/(FA - EB) \end{bmatrix}$$

İki doğru paralel ise, FA-EB=0 olur. Bu durumda doğrular kesişmezler.

Bilgisayar hesaplamalarını kolaylaştırmak için nokta koordinatları ile hesaplama yapılmalıdır. Şekil (4.6) 40'daki gibi birinci doğru parçalarının başlangıç ve bitim noktaları i(x_i, y_i) ve j(x_j, y_j), ikinci doğru parçasının başlangıç ve bitiş noktaları k(x_k, y_k) ve m(x_l, y_l) olsun.



Şekil (4.6) 40 i-j ve k-l doğrularının kesişimi

Doğru parçalarının paralel olup olmadığını belirlemek için d parametresi hesaplanır.

$$d = (x_i, y_i)(x_j, y_j) - (x_k, y_k)(x_l, y_l)$$

d=0 ise doğru parçaları paraleldir ve hesaplanamaz. $d \neq 0$ ise p_1 ve p_2 parametreleri hesaplanır.

$$p_1 = \frac{(x_k - x_l)(y_i - y_k) - (x_i - x_k)(y_k - y_l)}{d}$$

$$p_2 = \frac{(x_i - x_k)(y_j - y_i) - (x_j - x_i)(y_i - y_k)}{d}$$

$0 \leq p_1 \leq 1$ ve $0 \leq p_2 \leq 1$ ise kesişim noktası doğru parçalarının üzerindedir. Kesişim noktasının koordinatları;

$$x_s = x_i + p_1(x_j - x_i)$$

$$y_s = y_i + p_1(y_j - y_i)$$

Kesişim problemi programlama tekniği açısından düşünülürse reel sayılar olan nokta koordinatlarının hangi değişken tipi ile tanımlandığı önemlidir.

Bu algoritma kütüphanede doğru sınıfında kullanılmıştır.

4.5 Kutu Testi

Kutu Testi programlama tekniğinde çizgisel ve alansal nesnelerin birbiri ile olan ilişkilerini sınamakta kullanılan bir yöntemdir. Bir nesnenin dışına çizilen ve nesneyi tamamen kapsayan en küçük dikdörtgenlerin nesnelerin kendilerinden önce sınanması ile performans artışı sağlanır. Eğer Şekil (4.7) 41'de olduğu gibi i-j ile k-l doğru parçalarının kesişme durumları söz konusu ise;

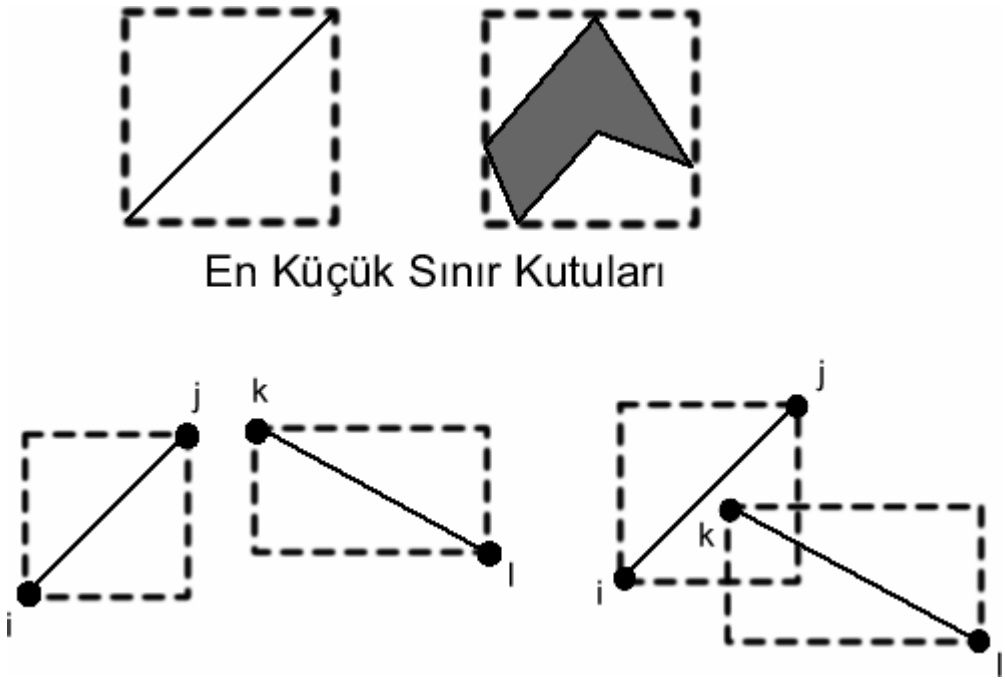
EĞER EnBüyük_(y_i, y_j) < EnKüçük(y_k, y_l) VE EnBüyük (x_i, x_j) < EnKüçük (x_k, x_l) ise

Kesişim mümkün değil

DEĞİLSE

Kesişme mümkün

Bu algoritma ile nesnelerin kendilerini sınamadan en küçük sınır kutuları sınanmakta ve kesişmeyen nesnelerde performans artışı görülmektedir [9] .



Şekil (4.7) 41 Kutu testleri

Bu algoritma kütüphanede kapsayan dikdörtgen (Envelope) sınıfında kullanılmıştır.

4.5 Nokta Poligonun İçinde mi?

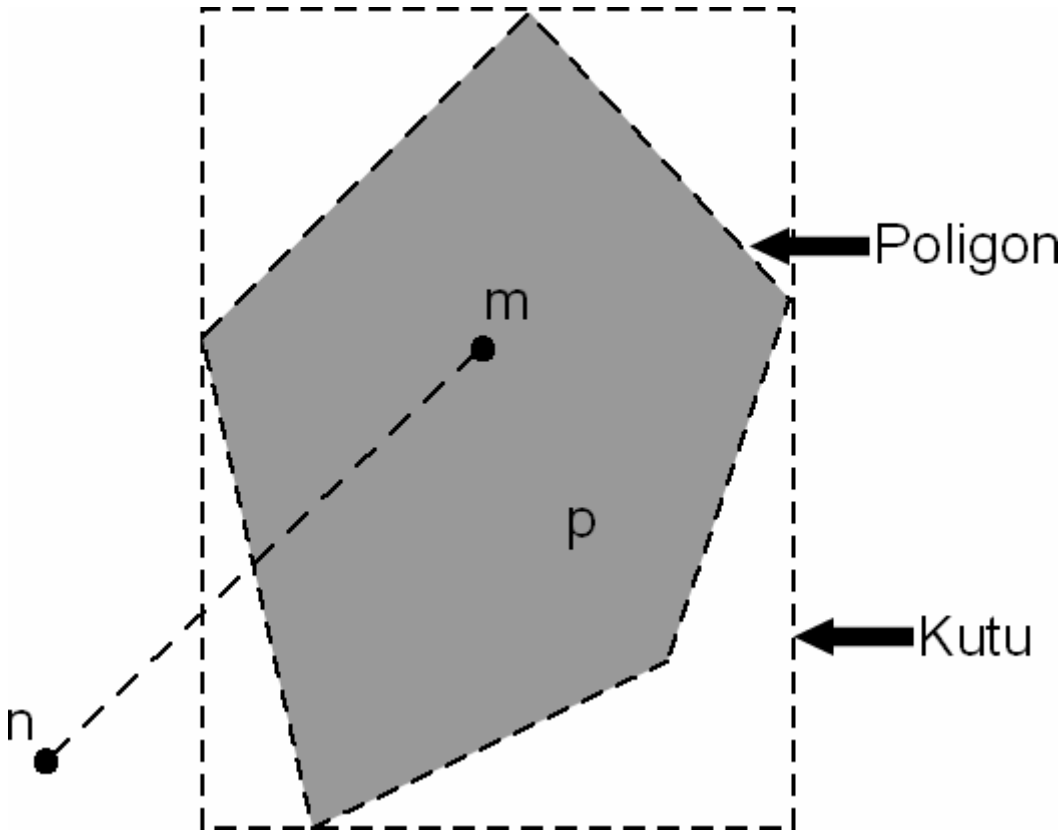
"Nokta Poligonun İçinde mi?" (*Point in Polygon*) Testi, bir noktanın bir alansal objenin içinde olup olmadığının belirlenmesi problemidir. Bir noktanın bir alansal objenin (kapalı şeklin) içinde olup olmadığını belirlemek için kapalı şeklin kesinlikle dışında olan bir yardımcı noktadan yararlanılır. Araştırılan nokta m, yardımcı nokta n olmak üzere m-n doğru parçasının, şekli kaç defa kestiği belirlenir. Kesişim sayısı tek ise nokta şeklin içinde, çift ise dışındadır [9] . *Nokta Poligonun İçinde mi* testi kendi içinde son derece basittir. Ancak arkasında programlama açısından çok da basit olmayan kesişim problemi vardır. Burada kesişim noktasının koordinatları ile değil yalnızca kesişimin var olup olmadığı ile ilgilenilmektedir.

Kesişim sayısının araştırılmasından önce, Şekil (4.8) 42'deki m noktasının P poligonu içinde olup olamayacağının incelenmesi gerekir. m noktası poligonu oluşturan kutunun içinde değilse poligonun da içinde olamaz. Kutunun sol alt köşe

koordinatları poligonu oluşturan nokta koordinatlarının en küçükleri, sağ üst köşe koordinatları ise şekli oluşturan koordinatların en büyükleri olduğuna göre;

Eğer enKüçük(x_i) > x_m > enBüyük(x_i) VE enBüyük(y_i) > y_m > enBüyük(y_i)

Şartı sağlanmıyorsa, nokta poligonun içinde olamaz. Şart sağlanıyorsa kesişim hesaplarına geçilebilir [2] .



Şekil (4.8) 42 Poligon, m noktası ve dışarıdan bir n noktası

Yardımcı n noktasının koordinatları, poligonu oluşturan kutu'dan yararlanarak aşağıdaki gibi hesaplanabilir:

x_n = enKüçük(x) - (enBüyük(x) - enKüçük(x))

y_n = enKüçük(y) - (enBüyük(y) - enKüçük(y))

Yukarıdaki eşitlik ile n noktasının kesin olarak poligonun dışında olması sağlanabilir. Uzakta bir n noktası seçme yerine eksenlerden birine paralel bir yardımcı doğru da seçilebilir [9] .

Poligonu oluşturan kenar sayısı kadar kesişim hesabı yapılarak, kesişme sayısı belirlenmek zorundadır. Bu aşamada da Bölüm 4.5 Kutu Testi'nde değinildiği gibi kesişim öncesi kutu testi uygulanarak, m-n doğrusu ile ilgili kenarın kesişme olasılığı var ise, d parametresi ile paralellik araştırılır.

Paralel olma durumu yok ise p_1 ve p_2 parametreleri hesaplanır. Eğer,

$$0 > p_1 > 1 \text{ VE } 0 > p_2 > 1$$

şartı sağlanıyorsa, kesişim var, aksi halde yoktur. Kesişim noktasının koordinatlarına ihtiyaç olmadığından hesaplama bu noktada kesilir, kesişim sayısı bir artırılır ve bir sonraki kenara geçilir.

Poligonu oluşturan tüm kenarlar için kesişim olup olmadığı araştırıldıktan sonra, toplam kesişim sayısı tek ise nokta poligonun içinde, çift ise değildir. Bu noktada bir üçüncü olasılık noktanın poligonu oluşturan kenarların birinin üzerinde olmasıdır. $0 > p_1 > 1$ VE $0 > p_2 > 1$ bağıntısına göre nokta herhangi bir kenar üzerinde ise ya da nokta poligonu oluşturan noktalardan biri ile çakışık ise noktanın poligon dışında olduğu kararı verilmektedir. Çünkü nokta kenarlardan biri üzerinde ise p_1 ya da p_2 sıfıra ya da bire eşit olur. Nokta poligonu oluşturan noktalardan biri ile çakışık ise hem p_1 hem de p_2 sıfıra ya da bire eşit olur. Bu durumlarda nokta poligonun içinde kabul edilecekse $0 > p_1 > 1$ VE $0 > p_2 > 1$ bağıntısı aşağıdaki gibi olmalıdır.

$$0 \geq p_1 \geq 1 \text{ VE } 0 \geq p_2 \geq 1$$

p_1, p_2 ve d parametreleri reel sayılar olduğundan, değişken tipi olarak yüksek hassasiyet kullanılması gerekir. Ancak bu şekilde tanımlanan değişkenlerin tam olarak sıfıra ya da bire eşit olmaları beklenemez. Sıfıra ya da bire eşit olma

bağıntısındaki mantık ile yapılmalıdır. Buradaki ε parametresi dikkatli seçilmelidir. Buradaki parametreler uzunluk değil, orantıdır. Örneğin p_1 parametresinin birden farkı, kesişim noktasının kenarlardan birine ne kadar yaklaştığı (uzaklık olarak) hakkında bir fikir vermez.

p_1 ve p_2 parametreleri uygun şekilde test edilerek nokta poligon içinde mi (*point in polygon)* testi sonucu, nokta içerde, nokta poligon kenarı üzerinde ve nokta dışarıda olmak üzere üç değişik sonuç da elde edilebilir.

Bu algoritma kütüphanede nokta ve poligon sınıflarında kullanılmıştır.

Bölüm 5 - Birim Testleri

Geometri Sınıf Kütüphanesinin birim testleri, Vivid Solutions Şirketi tarafından temin edilmiştir. Tamamen geometri sınıf kütüphanesi sınamasına yönelik bu veriler *OpenGIS*® standartlarına uygundur. Fakat bu test dosyalarını okunup işlenmesi için de bir takım dönüşümler yapılmalıdır. Test verileri bize operasyonlar, nesneler ve sonuçlar sağlar. Bu test verilerinden okunan nesneler çalışma zamanında oluşturulmuş, operasyonlar çağırılmış ve yine çalışma zamanında beklenen sonuçlarla karşılaştırılmıştır. Kısaca sınama içinde küçük bir uygulama oluşturulmuştur.

Çalışma zamanında hangi nesnenin oluşturulacağı bilinemediğinden, test uygulamasının bu sınıfları dinamik bir yapıda oluşturması gerekmektedir. Bu sorunun çözümü yine sınıf kütüphanesinin nesne yönelimli yapısından faydalanılarak bulunmuştur.

Bölüm 3'de sınıf kütüphanesinin yapısında da değinildiği gibi tüm sınıflar *Geometri* sınıfından türemektedir. Yani Geometri sınıfı tüm diğer sınıfların s*uper* sınıfıdır. Nesne yönelimli programlama dillerinde çalışama zamanında süper sınıfı kullanarak diğer sınıflar oluşturulabilmektedir.

Bahsi geçen bu sınama verileri geometri kütüphanesinin tüm operasyonlarını zorlayıcı verilerle sınamaktadır. Sınama dosyalarının bir listesi EK-1'de verilmiştir.

5.1 Sınama Dosyaları

Sınama dosyaları XML dosyalarıdır ve herhangi bir yazı editörü ile oynanabilir. Bir Test dosyası bir ya da daha çok sınama durumlarından oluşmaktadır (Test Case). Bu sınama durumları da bir ya da daha çok sınama bilgisi içermektedir (test). Her sınama bir sınama operasyonu ve beklenen değerleri içermektedir.

```
<run>
	<desc>example</desc>
	<precisionModel type="FIXED" scale="1" offsetx="0" offsety="0" />
	<case>
		<desc>point in a polygon</desc>
		<a>POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))</a>
		<b>POINT (5 5)</b>
		<test>
			<op name="contains">true</op>
		</test>
		<test>
			<op name="contains" arg1="b" arg2="a">false</op>
		</test>
	</case>
	<case>
		<desc>two lines that cross</desc>
		<a>LINESTRING (0 0, 10 10)</a>
		<b>LINESTRING (0 10, 10 0)</b>
		<test>
			<op name="crosses">true</op>
		</test>
	</case>
</run>
```

Şekil (5.1) 43 Örnek Sınama Dosyası

Şekil (5.1) 43'de bir sınama dosyası örneği gösterilmektedir. Örnekte iki adet durum bulunmaktadır:

İlk durumum iki sınama bilgisi bulunmaktadır:

- Poligon A B noktasını içerir.
- B noktası A poligonunu içermez.

İkinci durumu bir sınama içermektedir:

- A ve B doğruları birbirlerini kesmektedirler.

Sınama operasyonlarının beklenen değerleri başka bir geometrik nesne de olabilir. Örneğin iki doğrunun kesişme noktaları (POINT (10 17).

5.2 XML Tarifleri

Sınama XML dosyasının yazım tarifleri BNF formunda *EK–2 Sınama XML Dosyaları Tarifleri* kısmında verilmiştir. Burada XML dosyasının anlamsal kurallarına değinilecektir.

Her sınama dosyası *run* etiketi ile başlar. Bir *run* bir ya da daha çok *test case* içerir. *Run*, *test* ve *test case* etiketleri açıklama etiketi *desc* içerebilirler.

Bir *test case* etiketi bir ya da iki geometri içerebilir (ör. LINE(20 30, 10 10) ve POINT(40 10)). Çoğu operasyon iki geometri gerektirirken (ör. Overlaps) bazı operasyonlar sadece bir operasyon gerektirmektedirler (ör. isSimple, convexhull, getBoundary).

Her sınama bir işlenecek bir operasyon ve beklenen sonuç değerlerini içerir. İşlenecek operasyona parametre olarak gidecek değerler de (arg1, arg2) operasyon etiketine özellik olarak eklenir. Varsayılan olarak arg1 ve arg2'nin değerleri A ve B etiketlerinin değerleridir.

Burada argüman ve sonuçlar birer Geometrik Sınıf Kütüphanesi nesnesidir. Örneğin LINE(3 9 , 6 12) fakat göstrim olarak **Well-Known Text** olarak adlandırılmış, geometrik nesnelerin değiş tokuşu için kullanılan bir standart kullanılmıştır. Bu standardı *OpenGIS*® ortaya koymuştur [33] .

Bölüm 6 - Sonuç

Bu tez kapsamında Coğrafi Bilgi Sistemlerinin çekirdeğinde mutlaka bulunması gereken geometrik sınıf kütüphanesi, “Microsoft .Net Framework” ile gerçekleştirilmiştir. Bu kütüphanenin “Microsoft .Net Framework” ve ilişkili teknolojileri kullanacak Coğrafi Bilgi Sistemlerinin geliştirilmesinde kullanılması beklenmektedir.

Konumsal verilen sınıflandırılması, kullanıcılar için faydalı ve anlamlı ilişkilerinin belirlenmesi ve bunların hayata geçirilmesi için ortaya çıkmış çalışmalar günümüzde hatırı sayılır olgunluğa kavuşmuştur. Bu nedenle bu kütüphanenin genel yapısının uzun yıllar değişmeden kullanılabileceği düşünülmektedir.

OGC’nin ortaya koyduğu sınıf kütüphanesinin sadece bir tasarım olduğu düşünülmemelidir. Geometrik sınıf kütüphanesi standartları Bölüm 2 ve Bölüm 3’de anlatılan çalışmaların sonucu olarak ortaya çıkmıştır. Bu çalışmaların anlaşılması ve özümsenmesi, sınıf kütüphanesinin geliştiricisi için vazgeçilmez bir ön koşuldur. İleride Geometrik Sınıf Kütüphanesi Standartlarının değişmeyeceği garanti edilemez, araştırmacılar daha iyi modeller ortaya atabilirler, ama elbette ki bu modeller eski modelleri taban alacak modeller ve teoremler olacaktır. Bu durumda hangi modeller, veri yapıları ve nesne uzayları üzerine geliştirilme yapıldığının iyi anlaşılması sadece geometrik sınıf kütüphanesi geliştiricileri için değil, tüm CBS geliştiricileri için bir şarttır.

Bu geometrik sınıf kütüphanesi Boyutsal Olarak genişletilmiş dokuz kesişim modeli temel alınarak geliştirilmiştir [4] . Bu model iki boyutlu nesneler ve aralarındaki ilişkileri sınıflandırmaktadır. Gelecekte bu kütüphanenin üç boyutlu bir sürümü kütüphanenin esnek yapısı sayesinde kolaylıkla geliştirilebilir. Bununla birlikte yeni doğabilecek ilişkisel ve operasyonel ihtiyaçlara cevap verebilmek için daha gelişmiş ilişki ve veri türü modelleri de geliştirilebilir.

Yine bu kütüphane kullanılarak konumsal sorguların işlenmesi sağlanacaktır. Bölüm 1 ‘de anlatılan Konumsal Sorgu Motorunun gerekli ayıklama işlemlerini gerçekleştirdikten sonra bu kütüphanenin nesnelerini sorgunun içeriğine göre tanımlayıp aralarındaki işlemleri gerçekleştirebilecektir.

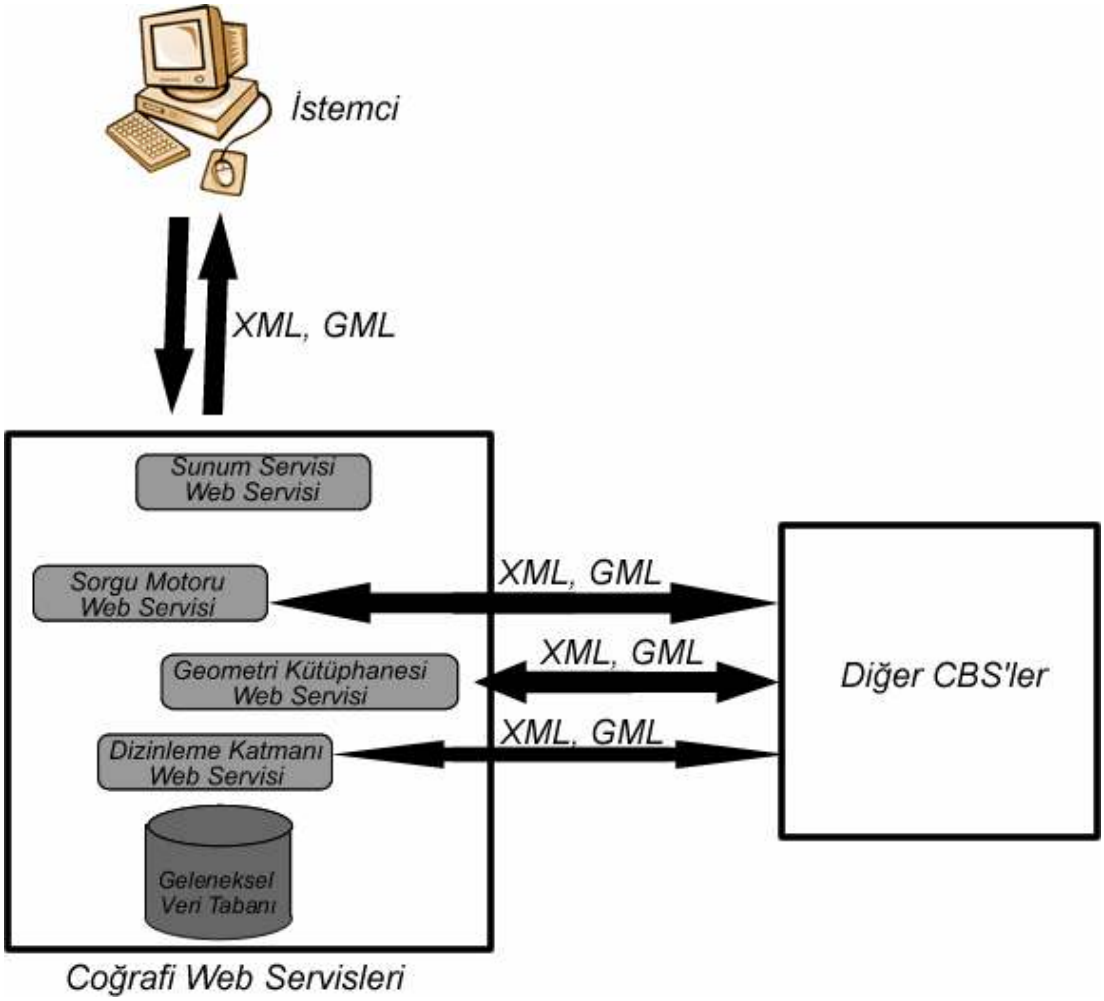
Geometrik Sınıf Kütüphanesi konumsal sorgu motorunun bir parçası olmasına rağmen dizinleme motoruna da bazı katkılarda bulunabilecektir. Örneğin dizinleme motoru cisimlerin en küçük sınırlayıcı dikdörtgeni bulma işini gerçekleştirirken yine bu sınıf kütüphanesinden yararlanacaktır. Yine dizinleme motoru gelen ya da değişen verileri veri işlerken bu geometrik sınıf kütüphanesinin nesneleri ile cisimleri tanımlayacak ve gerekli işlemleri yapabilecektir.

Konumsal verilerin geleneksel veri tabanlarında tutulmasının zorlukları ve geliştirilen çözümler Bölüm 1 de tartışılmıştır. Bulunan en iyi çözümün CBS geliştiricileri ile birlikte bağımsız OGC tarafından bulunan İyi Tanımlanmış Yazı standardı (WKT) ile olduğu açıktır. Bu standart konumsal verilerin veri tabanın bir alanında kolaylıkla tutulmasına olanak sağlamaktadır. Öyleyse bu İyi Tanımlanmış Yazı standardı ile veri tabanına koyulacak ve işlenecek verilerin bu formata getirilmesi ve yine bu formattan okunup tekrar konumsal nesneler haline getirilmesi gerekmektedir. Bu gerekliliği Konumsal sınıf kütüphanesinin Evliya-WKT adlı sınıfı gerçekleştirmektedir. Veri tabanına dizinleme ya da sorgu motorunun her erişimi sırasında mutlaka bu çevirici sınıf kullanılacaktır.

Geometri Sınıf Kütüphanesinin “Microsoft .Net” Platformunda oluşturulmasını temel nedeni, yazılan bu kütüphanenin web servisleri olarak hizmet verebilmesinin kolaylaştırılmasıdır. Gerçekten de “Microsoft .Net” Platformu’nda nesneleri hizmetlerini web üzerinden gerçekleştirmelerini sağlamak oldukça esnek ve kolaydır. Bu kütüphane, sorgu motoru ve dizinleme motoru gibi çekirdek parçaların ileride Şekil (5.2) 44’deki gibi web üzerinden hizmet verebileceklerdir. Yani Coğrafi Bilgi Sistemi ileride geleneksel bir uygulama olmaktan çıkıp, istemcilerin web üzerinden sorgu yapabilecekleri ve sonuçları bir internet tarayıcısından görüntüleyebilecekleri bir sistem olacaktır. Bu sistemin tüm parçaları (Geometrik Sınıf Kütüphanesi, Sorgu Motoru…) birbirlerinden bağımsız olarak çalışabilecekler ve veri alışverişlerini de

XML ve coğrafi verilerin taşınmasını sağlayan GML(Coğrafi İşaretleme Dili) ile gerçekleştireceklerdir.

Bu tez yazılırken sürmekte olan Evliya Çelebi Coğrafi Bilgi Katmanı TÜBİTAK-105K040 Projesinde ortaya çıkması beklenen sistem mimarisi Şekil (5.2) 44'de verildiği gibidir. Bu durumda "Microsoft .net" platformunda yazılmış bu kütüphane ileride web servislerine dönüştürülecek ve diğer katmanlara ve diğer CBS'lere hizmetlerini web üzerinden sunabilecektir.



Şekil (5.2) 44 Coğrafi Web Katmanları Sistem Mimarisi

Ek–1 Sınama Dosyaları

Aşağıda sınama verilerinin olduğu dosyalar ve açıklamaları bulunmaktadır.

- TestBoundary.xml – Tüm nesnelerin sınırlarını getiren fonksiyonlarını sınayan veriler.
- TestCentroid.xml – Tüm nesnelerin ağırlık merkezlerini bulan yöntemlerini sınama verileri.
- TestConvexHull.xml - Nesnelerin dış bükey kabuklarını bulan yöntemlerini sınayan veriler
- TestFunctionAA.xml – İki alan türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.
- TestFunctionAAPrec.xml
- TestFunctionLA.xml – Bir doğru ve bir alan türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.
- TestFunctionLAPrec.xml
- TestFunctionLL.xml - İki doğru türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.
- TestFunctionLLPrec.xml
- TestFunctionPA.xml- Bir nokta ve bir alan türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.
- TestFunctionPL.xml - Bir nokta ve bir doğru türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.
- TestFunctionPLPrec.xml
- TestFunctionPP.xml - İki nokta türündeki nesneler ile işlem yapan fonksiyonları sınayan veriler.

- TestInteriorPoint.xml
- TestRelateAA.xml – İki alan nesneleri arasındaki ilişkileri bulan metotları sınayan veriler.
- TestRelateAC.xml
- TestRelateLA.xml – Bir doğru ve bir alan türündeki nesnelerin aralarındaki ilişkileri bulan metotları sınayan veriler.
- TestRelateLC.xml
- TestRelateLL.xml - İki doğru nesneleri arasındaki ilişkileri bulan metotları sınayan veriler.
- TestRelatePA.xml - Bir nokta ve bir alan türündeki nesnelerin aralarındaki ilişkileri bulan metotları sınayan veriler.
- TestRelatePL.xml - Bir nokta ve bir doğru türündeki nesnelerin aralarındaki ilişkileri bulan metotları sınayan veriler.
- TestRelatePP.xml - İki nokta nesneleri arasındaki ilişkileri bulan metotları sınayan veriler.
- TestSimple.xml – Nesnelerin basit olup olmadığını hesaplayan metotları sınayan veriler.
- TestValid.xml – Nesnelerin geçerli olup olmadığını hesaplayan metotları sınayan veriler.
- TestWithinDistance.xml – isWithinDistance metodunu sınayan veriler.

EK–2 Sınama XML Dosyaları Tarifleri

testFile ::=

<run>

[<desc> text </desc>]

[<workspace dir=”directory”/>]

[<tolerance> double </tolerance>]

<precisionModel

type = “type” [scale="double" offsetx="double" offsetY="double"] />

{ caseText }

</run>

caseText ::=

<case>

[<desc> text </desc>]

<a> well-known-text | <a file=” filename”/>

 well-known-text | <b file=” filename”/>

{ testText }

</case>

testText ::=

<test>

[<desc> text </desc>]

<op name="opName"

arg1="geometry-index"

[arg2="arg"]

[arg3 | pattern ="arg"] >

result

</op>

</test>

opName ::=

equals | disjoint | intersects | touches | crosses |

within | contains | overlaps | relate | isSimple |

convexHull | intersection | union | difference | symDifference |

getBoundary| buffer | distance | getArea | isEmpty | isValid |

isWithinDistance | getEnvelope | distance | getLength

getDimension | getBoundaryDimension | getNumPoints | getSRID

result ::= boolean | integer | double | well-known-text

boolean ::= true | false

arg ::= geometry-index | null | integer | double | string

geometry-index ::= A | B

type ::= FIXED | FLOATING

EK—3 Sözlük

Dış Bükey Kabuk	Dış bükey kabuk uzayda bir şekli içine alan en küçük dış bükey çevre çizgisidir.
En küçük sınırlayıcı kutu (Minimum boundig box)	Bu sınıf geometrik nesnelerin iki boyutlu uzayda dikdörtgensel sınırlarını tanımlar
Geometrik Nesne İçi	Geometrik nesnenin aynı boyutlu sınırları hariç tüm koordinatları.
Geometrik Nesne sınırları	Bir geometrik nesnein sınırları kendinden bir boyut düşük olan ve onun sınırlarını oluşturan nesnelerdir.
Geometrik Nesne Tümleyeni	Bir geometrik nesnenin iç ve sınırları dışındaki tüm uzay koordinatları.
Konumsal Dizinleme (Spatial indexing)	Konumsal verilerin veri tabanına çeşitli yöntemlerle hızlı erişim için dizilmesi.
Konumsal SQL	Konumsal Yapısal Sorgulama Dili. Kullanıcıların konumsal özellikleri olan veri tabanlarından konumsal ve konumsal olmayan verileri sorgulayabilmelerini sağlayan sorgulama dili.
Microsoft .Net	Microsoft tarafından Windows makineler için geliştirilmiş bir yordamlar ve yorumlayıcılar kümesi.
Nokta Kümesi Topolojisi (Point-Set Topology)	Nokta Kümesi Topolojisi ya da Teorik Küme Topolojisi uzayda süreklilik ve kapalılık üzerine bir çalışma alanıdır.
Poligon (polygon)	Poligon birçok bağlı doğrudan oluşan düzlemdir.
SQL	Yapısal Sorgulama Dili. Kullanıcıların veri

	yığınlarından sorgulama yapabilmelerini sağlayan sorgulama dili.
Topoloji	Deformasyon sonucu değişmeyen nesne özelliklerini inceleyen bilim dalı.
XML	Genişletilebilir İşaretleme Dili, uygulamalar arası yalın yazı formatı kullanarak veri alışverişini kolaylaştıran World Wide Web Consorsium tarafından geliştirilmiş işaretleme dili.

Ek—4 Sorgu Motoru Aşamaları

Mantıksal Dönüştürüm:

SQL iç içe sorgu yapısını desteklediğinden, iç içe sorguları eşleniği (anlamı değiştirmeden dengini oluşturtmak) olan düz sorgulara çeviren algoritmalar SQL sorgu işlemcisinin önemli bir birimidir. Coğrafi-SQL önermeleri işlenen olarak alan alt sorguları desteklemediğinden, Coğrafi-SQL karşımıza normal SQL'den daha karmaşık iç içe sorgu yapısı çıkarmamaktadır. Bu nedenle normal SQL'in iç içeliği düz sorgulara çeviren algoritmaları ufak bazı değişikliklerle kullanılabilir. İç içe sorguları düz sorgulara çevirmek en iyileme işinin hazırlık aşaması olarak adlandırılabilir.

Konumsal olmayan bir önerme konumsal dizinleme yapısının kullanımını ileri aşamalarda engelleyebilir. Konumsal dizinlemenin kullanılabilmesi için bir sorgu anlamsal olarak eşleniği olan başka bir sorguya çevrilmelidir.

Aşağıdaki örneği inceleyecek olursak;

yol.touches(göl) and

(göl.usage = "Halka Açık" or

yol.isim = "E5").

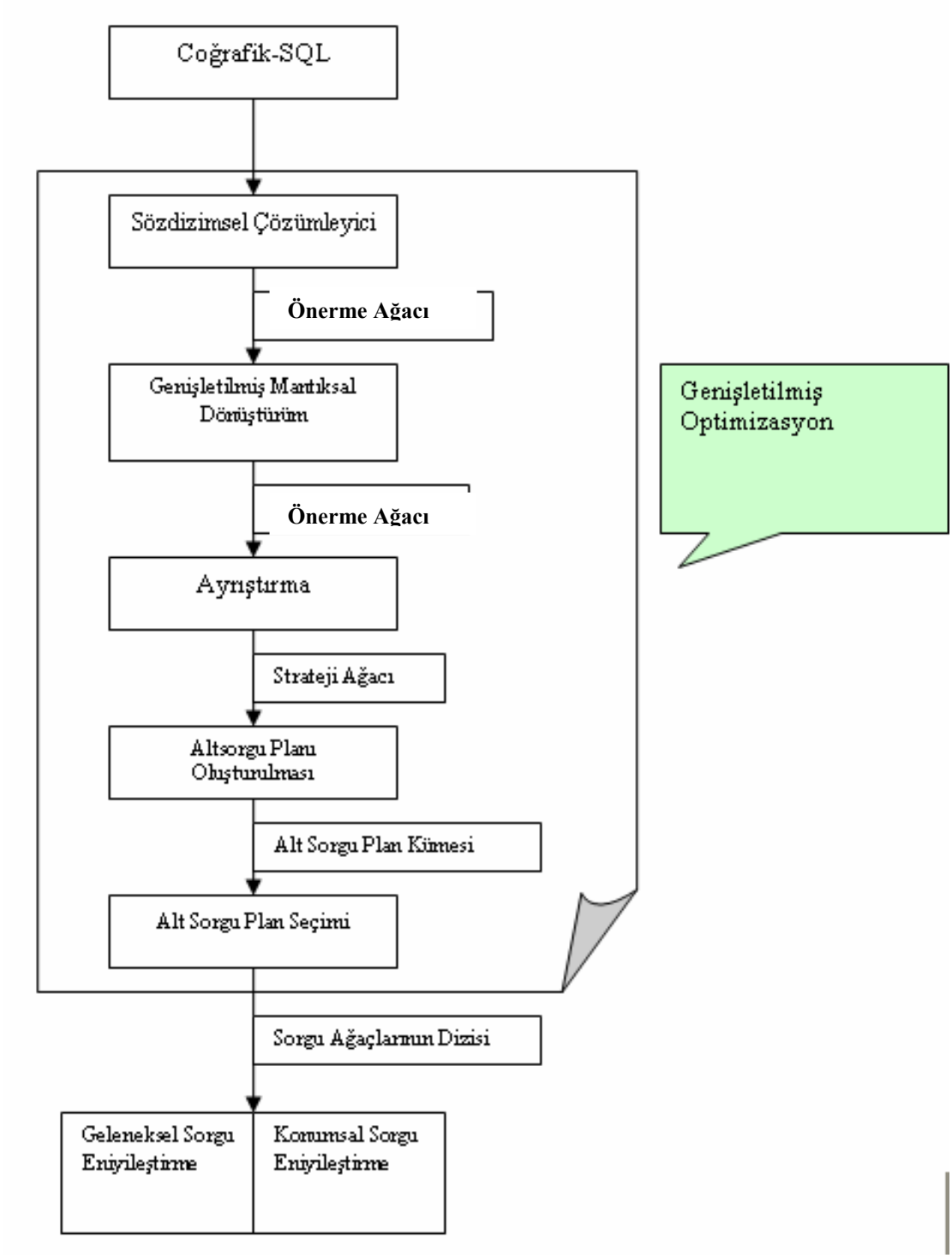
Eğer 'or' önermesi ilk önce işletilecek olursa, kartezyen çarpım (*and* işleci) konumsal dizinin kullanımını engelleyecektir. Bu durumda **and**'in **or**'a göre dağılımı özelliğini kullanarak aşağıdaki formun elde edilmesi daha uygun olabilir.

(yol.adjasent(göl) and

yol.isim = “E5”) or

(yol.adjasent(göl) and

göl.usage = “Halka Açık”)



Şekil (Ek-4.1) 45 Konumsal VT katmanı fonksiyonel mimarisi üzerinde genişletilmiş optimizasyon (en uygun şekle sokma)

Terim sayısını azaltmak sorgu en iyileştirmenin geleneksel bir kısmıdır. Mantıksal dönüştürme işlemlerinin en iyileştirilmesini, daha önce örnek üzerinde açıkladığımız

şekilde, işleçlerin (ve/veya önermelerin) cebirsel özelliklerini kullanarak gerçekleştirebiliriz.

Bu işlemin sonunda geleneksel VTYS'lerin ortaya çıkardıkları SQL sorgusu önerme ağacı ortaya çıkmaktadır. Ortaya çıkan bu ağaç yapısında, konumsal ve konumsal olmayan sorgular henüz iç içe bulunmaktadır.

Ayrıştırma:

Ayrıştırmada, önerme ağacı tamamı konumsal ya da konumsal olmayan alt ağaçlara ayrılır. Her alt ağaç bir alt bir alt sorguyu temsil etmektedir ve sorgu işlemcisi tarafından işlem görmelidir. Kısacası ayrıştırma, şekil 6'daki önerme ağacını, şekil 7'deki strateji ağacına dönüştürür. Bu ağacın içteki düğümleri mantıksal işlemleri tutan düğümler ve dıştaki düğümleri de alt sorgulardır. Strateji ağacının amacı, alt sorguların anlamlarını ve bağımlılıklarını sıraya dizilmeden yakalamaktır.

İşlem sonunda tüm konumsal önermeler ayrıştırılır, böylece konumsal işlemci konumsal önermeleri işleyebilir.

Sorgu ayrıştırmaya bir örnek verecek olursak;

```
SELECT yol.isim, demiryolu.isim
FROM   yol,alan,demiryolu
WHERE  yol.isim ≠ alan.isim and
           Yol. adjacent(alan) and
           demiryolu.isim ≠ 'Bölge 10' and
           (alan.isim = 'Başkent Üniversitesi' or
           alan.isim = 'Yenimahalle')
```

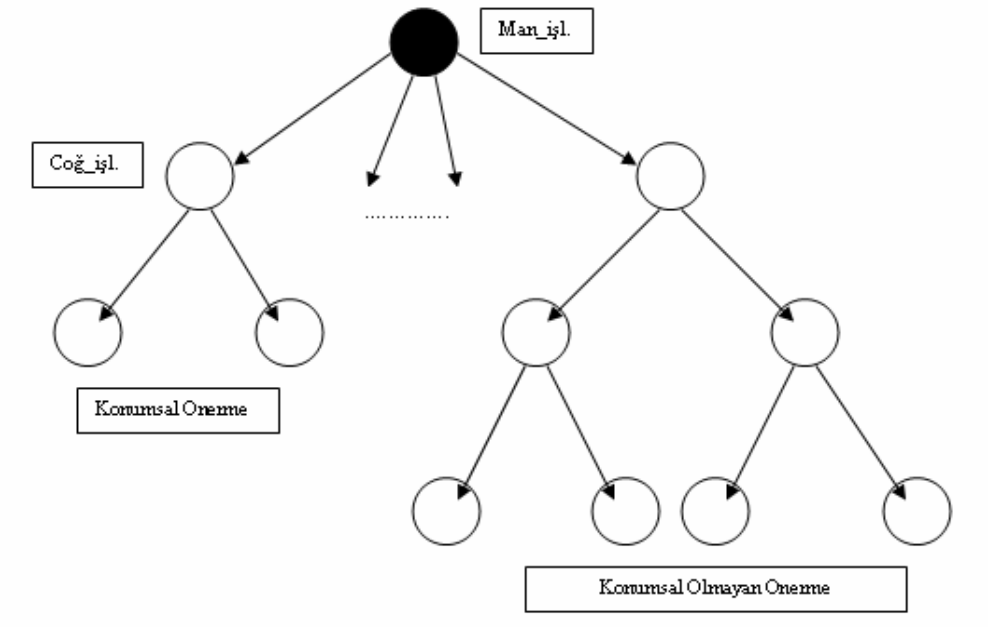
Yukarıdaki sorgudan strateji ağacı ile elde edilen dört alt sorgu aşağıdaki gibidir.

Q1: SELECT *
FROM alan,yol
WHERE yol.adjacent(alan).

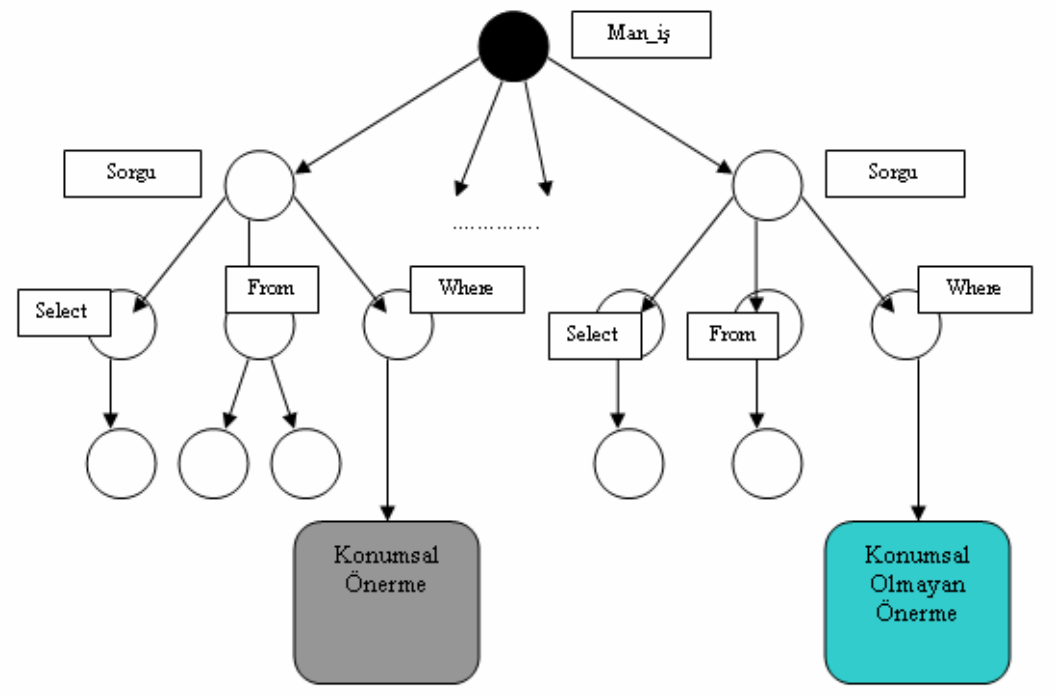
Q2: SELECT *
FROM yol,alan
WHERE yol.isim ≠ alan.isim.

Q3: SELECT *
FROM alan
WHERE alan.isim = ’Başkent Üniversitesi’ or
alan.isim = ‘Yenimahalle’

Q4: SELECT *
FROM demiryolu
WHERE demiryolu.isim ≠ ‘Bölge 10’



Şekil (Ek-4.2) 46 Önerme ağacı.



Şekil (Ek4.3) 47 Strateji ağacı.

Burada dikkat edilmesi gereken bir başka nokta da, ayrıştırma stratejisi sadece sorguyu alt sorgulara ayırmakta kullanılır, bize herhangi bir işlem sırası vermez. İşlem sırası belirlemek için plan formülasyonu ve seçim aşamaları gerekmektedir.

Plan Formülasyonu ve Seçim

Bir strateji ağacının alt sorguları sıralanmışsa, ağaç yapısından yakalanan bağımlılıklar fazladan SQL sorgularını ve/veya terimlerini yeniden yazma kısmında iyice ifade edilebilir. Birçok kısmi sonuç kümesini birleştirmek için (geçici ilişkiler) fazladan sorgu kullanılırken, terimleri yeniden yazma tekniği bir alt sorgunun kısmi sonucunu başka bir alt sorguya türetir.

Eğer ortak ilişkilere referans veren iki birleşmiş alt sorgu bağımsız olarak çalıştırılmışsa, kısmi sonuçlar doğal bitiştirme (natural join) işlemiyle birleştirilmelidir. Aşağıdaki örnek Q_a dan nasıl sonuç türetildiği ve fazladan Q_c sorgusu kullanılarak nasıl Q_b gibi çalıştırıldığı gösterilmektedir.

Q_a: SELECT *

FROM R_1, R_j

WHERE $P_1(R_i, R_j) and (P_2(R_i)$.

$T_b =$ SELECT *

FROM R_i

WHERE $P_2(R_i)$.

Q_b: SELECT *

FROM T_b, R_j

WHERE $P_1(T_2, R_j)$.

$T_{c1} =$ SELECT *

FROM R_i

WHERE $P_2(R_i)$.

$T_{c2} =$ SELECT *

FROM R_i, R_j

WHERE $P_1(R_i, R_j)$.

Q_c: SELECT *

FROM T_{c1}, T_{c2}

WHERE $T_{c1}.A_1 = T_{c2}.A_1 and T_{c1}.A_2 = T_{c2}.A_2$.

Bir önceki kısımda verilen Q1, Q2 ve Q3 örnek alt sorguları yukarıdaki sonuç yeniden yazma tekniğini kullanmalıdırlar. Q4 de ise diğer alt sorgular tarafından referans verilmese de sonuç birleştirmesiyle ya da başka bir alt sorgu ile son sonuç kümesine eklenmelidir (Q_c de olduğu gibi).

Eğer iki ayrık alt sorgu ayrık sonuç kümesine referans veriyorsa birleşim açık yapılmalı ve her kısmi sonuç referans vermedikleri ile kartezyen çarpıma girmelidir. Örneğin; S ayrık alt sorguların $Q_1 or ... or Q_k$ referans verdikleri ilişkiler kümesi

olsun. FROM cümlesinde referans verilen ilişkiler Q_i (i=1...k) S kümesinde ve T_i Q_i tarafından oluşturulan geçici ilişki olsun. Sonuç oluşturulma şekli:

$$T_1 \cup T_2 \cup .... \ T_k$$

Bölüm 7 Referanslar

[1] Barequet G. and S. Har-Peled, Efficiently approximating the minimum-volume bounding box of a point set in three dimensions, *J. of Algorithms (JoA), vol. 38 (1), pp. 91-109, January 2001.*

[2] Bildirici, İ.Ö., 2000, 1: 1000-1: 25 000 Ölçek Aralığında Bina ve Yol Objelerinin Sayısal Ortamda Kartografik Genelleştirmesi, *Doktora Tezi, İTÜ Fen Bilimleri Enstitüsü, İstanbul.*

[3] Chamberlin D. et al., "Sequel 2: A unified approach to data definition manupulation, and control," *IBM J. Res. Dev., vol. 20, no. 6, pp. 560-575, Nov. 1976.*

[4] Clementini Eliseo and Di Felice P., A Comparison of Methods for Representing Topological Relationships, *Information Sciences 80, 1-34, 1994.*

[5] Clementini Eliseo, Paolino Di Felice: Spatial Operators. *SIGMOD Record 29(3): 31-38 (2000)*

[6] Clementini, E., Di Felice, P., and van Oosterom, E A small set of formal topological relationships suitable for end-user interaction. *Proceedings of the Third International Symposium on Large Spatial Databases, Singapore, 1993.*

[7] Clementini, E., Di Felice, P.: A global framework for qualitative shape description. *Geoinformatica (1997)*

[8] Clementini, Eliseo, Di Felice, P., A Model for Representing Topological Relationships Between Complex Geometric Features in Spatial Databases, *Information Sciences 90 (1-4):121-136 , 1996.*

[9] Cromley, R.G., 1992, Digital Cartography, *Prentice Hall, New Jersey, 316p.*

[10] de Hoop, S. and van Oosterom, P. Storage and manipulation of topology in Postgres. *Proceedings of the Third European Conference on Geographical Information Systems, Miinchen, 1992.*

[11] Egenhofer M.. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering, 6:86-95, 1994.*

[12] Egenhofer, M. A formal definition of binary topological relationships. *Proceedings of the Third International Conference on the Foundations of Data Organization and Algorithms, Paris, 1989.*

[13] Egenhofer, M. and Frank, A. Towards a spatial query language: User interface considerations. *Proceedings of the Fourteenth International Conference on l,Very Large Data Bases, Los Angeles, 1988.*

[14] Egenhofer, M. and Herring, J. A mathematical framework for the definition of topological relationships. *Fourth International Symposium on Spatial Data Handling, Zfirich, 1990*

[15] Egenhofer, M., Frank, A., and Jackson, J.P. A topological data model for spatial databases. *Proceedings of the First International Symposium on Large Spatial Databases, Santa Barbara, 1989*

[16] Egenhofer, Max J.; Herring, John R. (1991): Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. *Technical report, Department of Surveying Engineering, University of Maine, Orono, ME, 1991.*

[17] Egenhofer. M.J. and Sharma, J., Topological Relations between regions in Â2 and Z2, *Advances in Spatial Databases—Third International Symposium, SSD '93, vol. 692, Lecture Notes in Computer Science, pp. 36-52, Springer Verlag, Singapore (1993).*

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

[19] Eliseo Clementini, Paolino Di Felice: An Object Calculus for Geographic Databases. *SAC 1993: 302-308*

[20] Frank A., "Requirements for a database management system for a GIS" *Photogrammetric Eng. & Remote Sensing, vol. 54, no. 11, pp.1557-1554, Nov. 1998.*

[21] Gargano, M., Nardelli, E., and Talamo, M. Abstract data types for the logical modeling of complex data. *Information Systems, 16:565-584, 1991.*

[22] Geographic Information Systems; *(Çevrimiçi: http://erg.usgs.gov/isb/pubs/ gis_poster/ , Erişim tarihi: 01.05.2005).*

[23] Graham R. L.. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters, 1:132--133, 1972.*

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

[25] Gültig R. H. An Introduction to Spatial Database Systems; *VLDB Journal,3, 357-399 (1994).*

[26] Güting, R.H. Geo-relational algebra: A model and query language for geometric database systems. In: Schmidt, J.W., Ceri, S., and Missikoff, M., eds., *Proceedings of the International Conference on Extending Database Technology, Venice, 1988.*

[27] Güting, R.H. and Schneider, M. Realms: A foundation for spatial data types in database systems. *Proceedings of the Third International Symposium on Large Spatial Databases, Singapore, 1993a.*

[28] Güting R.H , "Geo-relational algebra: a model and query language for geometric database systems," in Proc. Int. Cong. *Extending Database Technology, Venice, Italy, 1998, LNCS vol. 302. New York: Springer-Verlag, pp. 506-507.*

[29] Güting, R.H. GraphDB: A data model and query language for graphs in databases. *Fernuniversit/it Hagen, Informatik-Report 155, 1994*

[30] Güting, R.H. Second-order signature: A tool for specifying data models, query processing, and optimization. *Proceedings of the ACM SIGMOD Conference, Washington, DC, 1993.*

[31] comp.graphics.algorithms Frequently Asked Questions *(Çevrimiçi: http://www.faqs.org/faqs/graphics/algorithms-faq/ , Erişim tarihi: 09.02.2006)*

[32] Larue, T., Pastre, D., and Vi6mont, Y. Strong integration of spatial domains and operators in a relational database system. *Proceedings of the Third International Symposium on Large Spatial Databases, Singapore, 1993.*

[33] OGC Reference Modell; Open Geospatial Consortium Inc.; *Date: 2003-09-16; Reference number: OGC 03-040 ; Version: 0.1.3 (2005) (Çevrimiçi: http://www.opengis.org , Erişim Tarihi: 14.01.2005).*

[34] Ooi B.C., R. Sacks-Davis, and K.J. McDonell. Extending a dbms for geographic applications. *In Proceedings of 5th International Conference on Data Engineering, pp. 590--597, Los Angeles, California, IEEE Computer Society Press (1989).*

[35] Open GIS Consortium, Inc. *OpenGISÒ Simple Features Specification For SQL Revision 1.1 May 5*, 1999 (*Çevrimiçi: http://www.opengis.org Erişim Tarihi: 05.02.2005).*

[36] Rigaux P., Scholl M., A. Voisard; Spatial Databases With Application To GIS; *Morgan Kaufmann (2002) p. 410, ISBN: 1-55860-588-6.*

[37] Sadra N., "Extensions to SQL for historical databases," *IEEE Trans. Knowledge Data Eng., vol. 2, no. 2, pp. 220-230, June 1990.*

[38] Scholl, M. and Voisard, A. Thematic map modeling. *Proceedings of the First International Symposium on Large Spatial Databases, Santa Barbara, 1989.*

[39] Worboys, M.E A generic model for planar geographical objects. *International Journal of Geographical Information Systems*, 6:353-372, 1992.

ÖZGEÇMİŞ

Adı : Murat Soyadı : Hacıömeroğlu
Uyruğu : T.C.
Doğum Tarihi : 01/03/1979 Doğum Yeri : İstanbul

Şimdiki
Görev Yeri : Gazi Üniversitesi Bilgisayar Müh.
Görev Ünvanı : Araştırma Görevlisi
İş Adresi : Gazi Üniversitesi Mühendislik Mimarlı Fakültesi Bilgisayar Mühendisliği Bölümü Tandoğan – ANKARA
İş Tel. No : 0312 230 65 03 - 2134
e-posta : mhaciomeroglu@gazi.edu.tr

İLK VE ORTA ÖĞRENİM DURUMU

Okul	**İl / İlçe**	**Giriş**	**Çıkış**
Teğmen Kalmaz İlk. Öğr.	Ankara/Çankaya	1985	1990
A.A.A.L.	Ankara/Çankaya	1990	1997

YÜKSEK ÖĞRENİM DURUMU

Üniversite	**Ülke**	**Giriş**	**Çıkış**	**Ünvan**	**Derece**
Başkent Üniversitesi	Türkiye	1997	2002	Lisans	3.03
Başkent Üniversitesi	Türkiye	2003	2006	Y.L.	3.85

ÇALIŞTIĞI KURUMLAR

Kurum	**İl / İlçe**	**Giriş**	**Çıkış**	**Görevi**
SAS Bilişim	Ankara/Çankaya	2002	2003	Mühendis
Datasel Bil. Sis.	Ankara/Çankaya	2003	2003	Mühendis
Başkent Üniversitesi	Ankara/Etimesgut	2003	2006	Ar. Gör.
Gazi Üniversitesi	Ankara/Çankaya	2006	-	Ar. Gör.