

**YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

**KONUMSAL YETKİLENDİRME DESTEKLİ
VERİTABANI TASARIMI VE UYGULAMASI**

Harita Yük. Müh. Cüneyd HELVACI

**FBE Jeodezi ve Fotogrametri Mühendisliği Anabilim Dalı Uzaktan Algılama ve Coğrafi Bilgi Sistemleri
Programında Hazırlanan**

DOKTORA TEZİ

Tez Savunma Tarihi	: 09.07.2010
Tez Danışmanı	: Yrd. Doç. Dr. Mustafa Ümit GÜMÜŞAY (YTÜ)
Jüri Üyeleri	: Prof. Dr. Fatmagül BATUK Prof. Dr. Zübeyde ALKIŞ Prof. Dr. D. Zafer ŞEKER Prof. Dr. Nebiye MUSAOĞLU

İSTANBUL, 2010

İÇİNDEKİLER

	Sayfa
KISALTMA LİSTESİ	iv
ŞEKİL LİSTESİ	v
ÇİZELGE LİSTESİ	vii
ÖNSÖZ	viii
ÖZET	ix
ABSTRACT	x
1. GİRİŞ.....	1
2. COĞRAFI VERİLERİN DEPOLANMASI.....	3
2.1 Coğrafi Verilerin Depolanmasındaki Teknik Zorluklar	4
3. KONUMSAL VERİ İNDEKSLEME	6
3.1 R-Tree Konumsal Veri İndeksleme Yöntemi.....	6
3.1.1 R-Tree İndeks Kayıtları İçinde Arama Yapmak	7
3.1.2 R-Tree İndeks Kayıtlarına Ekleme Yapmak	8
3.1.3 R-Tree İndeks Kayıtlarından Birinin Silinmesi İşlemi.....	8
3.1.4 R-Tree İndeks Kayıtlarının Güncellenmesi.....	8
3.1.5 R-Tree Düzgümlerinin Bölünmesi	8
3.2 B-Tree Konumsal Veri İndeksleme Yöntemi.....	10
3.3 B-Tree ve R-Tree Konumsal İndeksleme Yöntemlerinin Karşılaştırılması	11
4. SQL SERVER 2005 İÇİN SİSTEM TASARIMI	12
4.1 Bağlayıcılar Modülü.....	12
4.2 Veritabanı Yönetim Modülü	14
4.3 Yetkilendirme Modülü	15
4.3.1 Yetkilendirme	15
4.3.2 Güvenlik İlkeleri.....	16
4.3.3 Erişim İlkeleri.....	16
4.4 İşlem Kuyruk Yöneticisi Modülü.....	17
4.5 Sorgu Modülü.....	18
4.6 Çevresel Veri Değiştirici Modülü	18
4.7 Sistem Yöneticisi Modülü	19
4.8 Konumsal Yetkilendirme Modülü.....	20
4.9 Konumsal İndeksleme Modülü	21
4.10 Konumsal Sorgu Modülü	22
4.11 Tasarımın Geliştirilmesi Aşaması	22
5. SQL SERVER 2008 YENİLİKLERİ	24
5.1 SQL Server 2008 Konumsal Veri Türleri	25
5.2 SQL Server 2008 Coğrafi Metodları	26
5.2.1 Coğrafi Veri Temsil Metodları	26

5.2.1.1	İyi Bilinen İkili Değer Yöntemi	27
5.2.1.2	İyi Bilinen Metin Yöntemi	29
5.2.1.3	Coğrafi İşaretleme Dili	31
5.2.2	Sayısal geometri komutları.....	34
6.	SQL SERVER 2008 İÇİN SİSTEM TASARIMI	37
6.1	Çekirdek Modül.....	37
6.2	Ayrık Tablolar Yöntemi	40
6.3	Yetki Alanı Yöntemi	41
6.4	Ayrık Tablolar ve Yetki Alanı Yöntemlerinin Karşılaştırılması.....	41
7.	SİSTEM YÖNETİM MODÜLLERİNİN GELİŞTİRİLMESİ.....	46
7.1	Uygulama Veritabanının Oluşturulması.....	46
7.2	Yönetim Tablolarının Oluşturulması.....	46
7.3	Test Tablolarının Oluşturulması.....	46
7.4	Veritabanı Metodlarının Geliştirilmesi	46
7.5	Yetki Yönetimi Alt Sisteminin Geliştirilmesi	47
7.5.1	Tabloya Ekleme İşleminin Yönetilmesi	47
7.5.2	Tablodaki Veriye Erişimin Yönetilmesi.....	48
7.5.3	Silme İşlemi Sonrasında Kayıtların Güncellenmesi.....	49
8.	UYGULAMANIN GERÇEK VERİLER İLE TEST EDİLMESİ	51
9.	PERFORMANS TESTLERİ.....	57
9.1	Ekleme Testi.....	58
9.2	Güncelleme Testi.....	59
9.3	Veri Silme Testi.....	59
9.4	Veri Okuma Testi	60
9.5	Performans Testleri Değerlendirmesi.....	60
10.	SONUÇLAR	62
	KAYNAKLAR.....	64
	EKLER	66
Ek 1	Program Kodları	67
Ek 2	Uygulama veritabanı SQL komutları	129
	ÖZGEÇMİŞ.....	136

KISALTMA LİSTESİ

SQL	Structured Query Language
T-SQL	Transact-SQL
GIS	Geographic Information System
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
XML	Extensible Markup Language
GUID	Globally Unique Identifier
IAM	Index Allocation Map
OGC	Open Geospatial Consortium
WKB	Well Known Binary
WKT	Well Known Text
GML	Geographic Markup Language
CBS	Coğrafi Bilgi Sistemi
CAD	Computer Aided Design

ŞEKİL LİSTESİ

Şekil 2.1	SQL geometri türlerinin hiyerarşisi	4
Şekil 2.2	Kesişim örneği üzerinde bilgi depolama çözünürlüğünün gösterimi	4
Şekil 2.3	Delikli bir poligonun iki ayrı nesne ile ifade edilmesi.....	5
Şekil 3.1	(a) R-Tree Araması (b) R-Tree grafiksel gösterimi	7
Şekil 3.2	Yeni bölümlleme sonrası oluşan yeni I elemanları	9
Şekil 3.3	Dört seviyeli grid hiyerarşisi.....	10
Şekil 3.4	Mozaikleme ile konumsal indeks kullanılarak yapılan arama	11
Şekil 4.1	Sistem modülleri	12
Şekil 4.2	Bağlayıcılar modülü use case diyagramı	13
Şekil 4.3	Veritabanı yönetim modülü use case diyagramı	14
Şekil 4.4	Yetkilendirme modülü use case diyagramı	17
Şekil 4.5	İşlem kuyruk yöneticisi modülü use case diyagramı	17
Şekil 4.6	Sorgu modülü use case diyagramı	18
Şekil 4.7	Çevresel veri değiştirici modülü use case diyagramı.....	19
Şekil 4.8	Sistem yöneticisi modülü use case diyagramı	20
Şekil 4.9	Konumsal yetkilendirme modülü use case diyagramı	20
Şekil 4.10	Konumsal indeksleme modülü use case diyagramı.....	21
Şekil 4.11	Konumsal sorgu modülü use case diyagramı	22
Şekil 5.1	Geometry ve geography veri türlerinin konumsal nesneler ile ilişkileri.....	25
Şekil 5.2	Coğrafi nesne türleri	26
Şekil 7.1	trgInsert_YetkiKontrol akış diyagramı	48
Şekil 7.2	{%Tablo Adı}_YETKILENDIRILMIS akış diyagramı	49
Şekil 7.3	trgInsert_YetkiKontrol akış diyagramı	50
Şekil 8.1	Microsoft SQL Server 2008 Management Studio ile yapılan ve sonucu coğrafi bir veri içeren temel bir sorgu ekran görüntüsü.....	51
Şekil 8.2	Test kullanıcısının tablo_Ulkeler üstündeki yetkileri	52
Şekil 8.3	Test kullanıcısının tablo_Ulkeler tablosuna erişimin kısıtlandığını gösteren hata mesajı	53
Şekil 8.4	Test kullanıcısının tablo_Ulkeler_YETKILENDIRILMIS() Fonksiyonu üstündeki yetkileri	54
Şekil 8.5	Test kullanıcısı tarafından çalıştırılan tablo_Ulkeler_YETKILENDIRILMIS() fonksiyonunun sonucu	55

Şekil 8.6 Test kullanıcısı tarafından çalıştırılan tablo_Ulkeler_YETKILENDİRİLMİS() fonksiyonunun yeni sonucu	56
Şekil 9.1 Performans testi için kullanılan veri	57
Şekil 9.2 Performans test uygulaması	57
Şekil 9.3 Veri ekleme uygulaması arayüzü	58
Şekil 9.4 Genel durum grafiği	61

ÇİZELGE LİSTESİ

Çizelge 3.1	Grid yoğunlukları.....	10
Çizelge 6.1	Konumsal yetkilendirme bilgilerini depolayan veritabanı tabloları	38
Çizelge 6.2	KYCM tablosuna ait alanlar	38
Çizelge 6.3	KYB_{Tablo Adı} tablosuna ait alanlar.....	38
Çizelge 6.4	KYBKI_{Tablo Adı} tablosuna ait alanlar	39
Çizelge 6.5	Konumsal yetkilendirme sistemi çekirdek modülü veritabanı prosedürleri	39
Çizelge 6.6	Veritabanı işletme süreci eylemleri için yöntemlerin karşılaştırılması.....	42
Çizelge 6.7	Veritabanı bakım süreci eylemleri için yöntemlerin karşılaştırılması	43
Çizelge 6.8	Veritabanı kurtarma süreci eylemleri için yöntemlerin karşılaştırılması.....	44
Çizelge 6.8	Veritabanı kurtarma süreci eylemleri için yöntemlerin karşılaştırılması.....	45
Çizelge 9.1	Veri ekleme test sonuçları.....	58
Çizelge 9.2	Veri güncelleme test sonuçları.....	59
Çizelge 9.3	Veri silme test sonuçları.....	59
Çizelge 9.4	Veri okuma test sonuçları	60

ÖNSÖZ

Bu tez çalışmasında, günümüzde gelişen bilgi teknolojileri kullanımında ihtiyaç duyulan konum bilgisine erişiminin her uygulama geliştirici tarafından kolaylıkla denetlenebilmesi için bir uygulama tasarlanmış ve geliştirilmiştir.

“Konumsal Yetkilendirme Destekli Konumsal Veritabanı Tasarımı ve Uygulaması” konulu doktora tezimin yürütücülüğünü üstlenen ve tezimin her aşamasında bana yol gösteren, yardım ve desteğini esirgemeyen hocam sayın Yrd. Doç. Dr. M. Ümit GÜMÜŞAY’a teşekkürlerimi sunarım.

Çalışmalarım sırasında destek ve yardımlarını esirgemeyen tez izleme komitesi üyeleri, Prof. Dr. Fatmagül BATUK ve Prof. Dr. Dursun Zafer ŞEKER’e teşekkürü borç bilirim.

Çalışmalarım sırasında destek ve yardımlarını esirgemeyen Fotogrametri ve Uzaktan Algılama Anabilim Dalı hocalarıma ve Araştırma Görevlisi arkadaşlarıma teşekkürü borç bilirim.

Bu çalışmayı gerçekleştirdiğim süre içinde yardımını, sevgisini ve ilgisini üzerimden eksiltmeyen eşim Yeşim’e ve tatlı gülücükleri için oğlum Mehmet Hakan’a teşekkürler.

Çalışmalarım süresince bana 1400 km öteden varlıklarıyla her türlü desteği veren anneme ve kız kardeşlerime teşekkürler.

17 senelik beraberliğimiz süresince bitmek bilmez sorularıma bıkmadan cevap vererek araştırmacı ve meraklı tarafımı geliştiren, öğrenmeyi öğreten ve sevdiren insana; babam Ahmet Remzi HELVACI’ya sonsuz teşekkürler.

ÖZET

Bir kentteki kent bilgi sistemi düşünöldüğünde, coğrafi veritabanlarındaki veriler farklı kurumlara ve/veya coğrafi alanlara aittir. Birçok veritabanında yeralan aynı coğrafi bölgeye ait farklı kayıtlara erişim isteğı günümüzde çok artmıştır. Güncel ve doğru veriye bu şekildeki bir paylaşım ile erişim ancak veritabanlarının birleştirilmesi ve erişimi coğrafi alanlar ile sınırlamakla mümkündür.

Bu çalışmanın amacı;kullanıcının coğrafi veritabanındaki verilere erişimini, önceden tanımlanmış coğrafi yetki alanı ile sınırlamaktır. Bu amaçla Konumsal yetkilendirme problemi araştırılmış, örnek bir çözüm tasarlanmış, geliştirilmiş ve karşılaştırma testleri yapılmıştır.

Çalışma platformu olarak, başlangıçta Microsoft SQL Server 2005 için tasarım yapılmış, daha sonra Microsoft SQL Server 2008 için yeniden tasarım yapılmış, T-SQL ve .Net Framework - C# programlama dilleri ile sistemi geliştirilmiştir.Geliştirilen sistem ile veritabanı kullanıcılarının erişim yetkileri coğrafi alanlara göre sınırlandırılmıştır.

Avrupa nehirlerini içeren veriler, Microsoft SQL Server 2008, Oracle 11g ortamında okuma, ekleme, silme, düzenleme işlem süreleri için test edilmiş ve sonuçlar karşılaştırılmıştır. Geliştirilen sistemin; okuma yoğun sistemler için daha uygun olduğı gözlenmiştir.

Geliştirme işleminin ardından sistemin testleri yapılmış ve eksiklikleri giderilmiştir. Ayrıca geliştirilen sistemin Microsoft SQL Server 2008 ve Oracle 11g ile arasındaki performans farkları da örnek veri ile test edilmiştir.

Anahtar Kelimeler: Veritabanı, coğrafi veritabanı, veri kalitesi, veri güvenliğı, konumsal yetkilendirme, konumsal veri indeksleme

ABSTRACT

When the information system of a city is considered, the data in geographical databases belong to various institutions and/or geographical areas. Requests for access to various records of same geographical areas that are available in many databases have significantly increased recently. However, the access to updated and correct data by the users from different institutions can only be achieved by uniting databases and limiting access to geographical areas.

The purpose of this study is to limit access to data in geographical databases to predefined geographical scopes of authority. The question of positional authorization has been looked into with this design and an exemplary solution has been designed and developed. Additionally, required comparison tests have been conducted.

At the beginning, a design was devised for Microsoft SQL Server 2005 as the working platform, later a new design was devised for Microsoft SQL Server 2008, and the system was developed with T-SQL and .Net Framework - C# languages. With the system devised, access authorization of database users was limited according to geographical areas.

The data concerning European rivers were tested for reading; adding, deleting and editing process times in Microsoft SQL Server 2008, Oracle 11g environments and the results were compared. It was observed that the developed system has been more suitable for read-intense systems.

After the process of development, the system was tested and the deficiencies were corrected. On the other hand, performance differences of the system developed were tested in Microsoft SQL Server 2008 and Oracle 11g by using the sample data.

Key Words: Database, spatial database, data quality, data security, spatial authorization, spatial data indexing

1. GİRİŞ

Veriyidepolamakisteyen herkesin karşılaşacağı temel problemlerden biri verinin güvenliğidir. Güvenlik,verinin paylaşımını yapan her sistemin tasarım, geliştirilme ve işletme süreçlerinin hepsinde ihmal edilemeyecek kadar önemli bir parametredir.

Bugün ki ticari ve kamusal uygulamaların kullanıcı sayıları milyonlarla ifade edilmektedir. Bu kadar büyük kullanıcı sayısı e-devlet uygulamaları gibi kişiye özel bilgileri de barındıran uygulamalarda küçük bir güvenlik boşluğunun tesadüf ile de keşfedilebileceği anlamına gelir.Güvenlik açıklarının sonuçları sadece veri hırsızlığı ile sınırlı kalmayacaktır. Güvenliği sağlanmamış bir verinin kötü niyetli bir kullanıcı tarafından çalınması, silinmesi veya değiştirilmesi ile veri sahibinin uğrayacağı zarar çok büyük olabilir. Bu işlemin kasıtsız olarak yapılması zararı azaltmayacaktır.

Bugün kullanımda olan veritabanı sunucularının tamamı çeşitli yöntemlerle kullanıcıların veriye olan erişimini kontrol altında tutarlar. Ancak hiçbir veritabanı sunucusu kullanıcılarının kayıtlara olan erişimini coğrafi alanlar ile sınırlandırmamaktadır.

Yıllarca coğrafi veri ayrı ayrı dosyalarda depolanarak organize edilmiştir (Zlatanova, 2006).Veritabanı araştırmacılarının çoğu ofis uygulamaları, Coğrafi Bilgi Sistemleri (CBS), Bilgisayar Destekli Tasarım (CAD) vb. standart olmayan uygulamalara destek verebilmek için çalışmalar yapmaktadır (Güting, 1989). Bu çalışmalar sonucunda eskiden sadece sözel verileri depolayan veritabanları artık coğrafi verileri de depolamaktadır.

Veritabanı yönetim sistemleri coğrafi nesneleri yönetecekşekilde tasarlanmamıştır, fakat ilişkisel matematik ve hesaplama üzerine kurulu fonksiyonlar içerir (Ramakrishnan, 2002). Bu fonksiyonlar ile konumsal veri oluşturma ve değiştirme işlemleri gerçekleştirilir.

Konumsal yetkilendirme, veritabanı içindeki kayıtların coğrafi konumuna göre tanınarak kullanıcıların erişimlerinin denetlenmesidir. Bunun için her kaydın bir coğrafi ilişkisinin bulunması gereklidir. Buna göre coğrafi konumu belirsiz olan bir verinin konumsal yetkilendirme ile korunması da söz konusu değildir.

Bu çalışmada; veritabanı içindeki kayıtlara erişimin kullanıcılar için tanımlanmış olan yetki bölgeleri ile sınırlandırılması ile pratikte yaşanan birçok probleme çözüm getirmesi amaçlanmıştır.

Konumsal yetkilendirme ihtiyacı için uygulama geliştiricilerin temel ve en çok kullandıkları teknik veritabanlarını ayırmaktır. Bu tekniğin beraberinde getirdiği problemler vardır.

Bunlardan en önemlisi veri tekrarıdır. Örneğin belediyeye ait bir veritabanı ile muhtara ait veritabanı kesişen bölgelere ait verinin iki yerde bulunması anlamına gelir ve hangisindeki bilginin doğru olduğu bilinmemektedir. Aynı veri İller Bankasında ve Emniyet Müdürlüğünde de bulunduğu düşünülürse, aynı veri 4 ayrı veritabanında bulunmaktadır. Günlük hayatımızda karşımıza çıkan birçok problemin sebebi bu dağınıklıktır.

Bu kurumlar tüm verilerini tek bir veritabanı içinde depolandıklarında ise yetki karmaşası oluşmaktadır. Bu karmaşanın çözümü konumsal yetkilendirmedir.

Konumsal yetkilendirme, veritabanı ve uygulama içinde olmak üzere iki farklı şekilde uygulanır. Ancak uygulama içinde yapılan konumsal yetkilendirme gerçek bir erişim yetkilendirmesi değildir.

Verinin daha veritabanından çıkmadan süzülmesi ve yetkisi olmayanlara ulaştırılmaması gerekmektedir. Bu yöntem beraberinde birçok fayda getirir. Örneğin mahalle verisine erişen muhtar her defasında tüm şehrin verisini çekmek yerine sadece yetkili olduğu veriyi yani kendi mahallesine ait verileri ağ üstünden aktaracaktır. Bunun muhtara faydası zaman; belediyeye (veritabanını barındıran kurum) faydası ise ağ kapasitesinin daha az kullanılmasıdır. Bu sayede sistem muhtar ve belediye daha hızlı hizmet verebilecektir. Bunlar görünen faydalardır. Görünmeyen faydalar ise özetle üçüncü parti bir veri sunucu yazılımına gerek kalmadığı için veriye erişimin standart hale gelmesi ile gelişmiş bağlantılabilirlik ve veritabanı sunucusunun tüm özelliklerinden yararlanabilirliktir.

Sistemigerçekleştirmek için tasarıma 2006 yılında Microsoft SQL Server 2005 ile başlanmıştır. Ancak 2007 yılının Kasım ayında Microsoft tarafından Microsoft SQL Server 2008'in "geometry" ve "geography" olmak üzere iki konumsal veri türünü destekleyeceği duyurulmuştur. Microsoft SQL Server 2008 sürümünün, 2005 özelliklerine olan desteği ve yeni özellikleri araştırılmıştır. Sonuçta 2005'e tam uyumluluğun mümkün olmadığı görülmüş ve tasarım Microsoft SQL Server 2008 temel alınarak yenilenmiş, bileşenler T-SQL ve .Net Framework - C# programlama dilleri ile geliştirilmiş, hata ayıklama testi aşamasında sonra gerçek veriler ile uygulama yapılmıştır.

Sistemin performans testleri için yaklaşık seksen sekiz bin kayıt içeren Avrupa nehirleri coğrafi verileri üç ayrı veritabanında depolanmıştır. Bu veritabanları;

- Microsoft SQL Server 2008
- Microsoft SQL Server 2008 (Konumsal Yetkilendirmeli)
- Oracle 11g

2. COĞRAFI VERİLERİN DEPOLANMASI

Coğrafi verilerin depolanmasından önce bunların tanımlanması ve ilişkilerinin belirlenmesi gereklidir. Konumsal veritabanında depolanacak, sorgulanacak ve analiz edilecek olan coğrafi nesnelerin özelliklerinin ve ilişkilerinin tasarım öncesinde tüm ayrıntıları ile ortaya koyulması gerekmektedir. Bu yüzden OGC (OpenGIS Consortium, Inc.) tarafından 5 Mayıs 1999 tarihinde yayınlanan “OpenGIS Simple Features Specification For SQL Revision 1.1” dokümanında belirtilen detaylar tasarım ve uygulama aşamasında doğrudan kullanılmıştır.

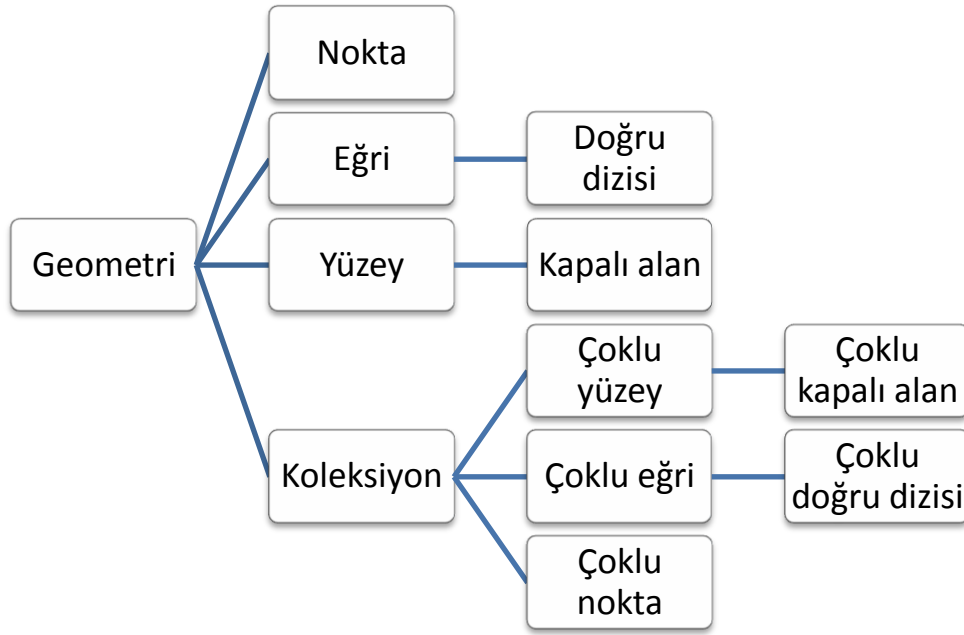
Coğrafi veriler temel olarak üçe ayrılırlar. Bunlar;

- a. Nokta
- b. Eğri
- c. Yüzeydir.

“OpenGIS Simple Features Specification For SQL Revision 1.1” dokümanı bu nesneler arasındaki tüm ilişkileri ve bunlara uygulanacak olan fonksiyonları düzenler. Bu dokümana göre SQL geometri türlerinin hiyerarşisi Şekil 2.1’de gösterilmiştir. Buna göre OGC standartlarını kabul eden bir sistemin destekleyeceği minimum veri türleri,

- Geometri (Geometry)
- Nokta (Point)
- Eğri (Curve)
- Doğru dizisi (Linestring)
- Yüzey (Surface)
- Çokgen (Polygon)
- Koleksiyon (Collection)
- Çoklu nokta (Multipoint)
- Çoklu eğri (Multicurve)
- Çoklu doğru dizisi (Multilinestring)
- Çoklu yüzey (Multisurface)
- Çoklu çokgen (Multipolygon)

Şeklinde olmalıdır.

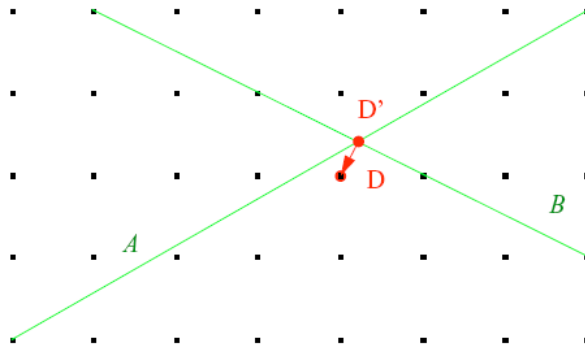


Şekil 2.1 SQL geometri türlerinin hiyerarşisi (OpenGIS Project Document 99-049, 1999)

2.1 Coğrafi Verilerin Depolanmasındaki Teknik Zorluklar

Bilgisayarın doğasından gelen bazı kapasite zorlukları mevcuttur. Bilgisayar ortamında depolanan tüm verilerin aslında sayılardan oluştuğu ve temelinde 1 ve 0'lerden meydana geldiği bilinmektedir. Bilgisayar hafızasında, her bilgi belirli bir sayısal çözünürlükte depolanmaktadır. Çözünürlük arttıkça daha çok alana ihtiyaç vardır. Bu ilişkiden dolayı çözünürlük belirlenirken dikkatli davranılmalıdır.

Coğrafi verilerin depolanmasında çözünürlüğün etkisi Şekil 2.2'de örneklenmiştir. Şekil 2.2'de A ve B doğrularının gerçek kesişim noktası D' ile, bilgisayar tarafından depolanan nokta ise D ile gösterilmektedir. Buna göre kesişim D' iken bilgisayar bunu D olarak hesaplar ve depolar.

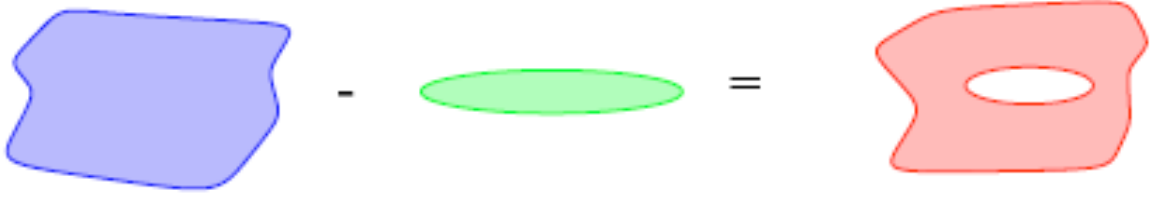


Şekil 2.2 Kesişim örneği üzerinde bilgi depolama çözünürlüğünün gösterimi

Çözünürlüğün uygulamadaki bu etkisi hatalara sebebiyet verebilmektedir. Özellikle çalışmanın konusu olan yetkilendirme bölgelerinin sayısal ifadesiningerçek yetki alanından farklı olacağı bilinmektedir.

Çözünürlükten kaynaklanan bu ve benzeri sorunlar tasarım aşamasının her adımında ayrı ayrı irdelenmiş ve tasarım buna göre yapılmıştır.

Diğer bir teknik sorun ise karmaşıklık sorunudur. İlişkisel veritabanı içinde nokta veya çizgi gibi bir nesneyi depolamak ve sorgulamak basittir. Ancak karmaşık şekillerin depolanması ve sorgulanması zordur ve ek yöntemler gerektirir. Şekil 2.3’de gösterilen delikli bir poligon tek başına bir obje olarak depolanamaz. Ancak birden fazla nesne ve bu nesnelerin ilişkileri ile depolanabilir.



Şekil 2.3 Delikli bir poligonun iki ayrı nesne ile ifade edilmesi.

3. KONUMSAL VERİ İNDEKSLEME

Konumsal verilere erişim hızının artırılması için kullanılan tekniğe “Konumsal Veri İndeksleme” denir. İndekslemeye en bilinen örnek olarak bir kitabın arkasındaki kelime sayfa ilişkilerini içeren indeks bölümü gösterilebilir. Coğrafi veriler için bu ilişkiler temel olarak nesneler ve küçük coğrafi pencereler şeklinde kurulmaktadır. Bu şekilde büyük bir veri yığını içindeki coğrafi bir objeye hızlı bir şekilde ulaşılabilir.

Veri indeksleme için geliştirilmiş birçok yöntem mevcuttur. Uygulama sırasında yapılacak olan sorgulamalara göre uygulanacak yöntemlerin bir veya birkaçı seçilir. Verilerin sayısal ortamda depolanmaya başlandığı günlerden bugüne kadar gelişim gösteren indeksleme yöntemleri konumsal verilerinde sayısal ortamda depolandığı günlerden itibaren gelişim göstermektedir.

3.1 R-Tree Konumsal Veri İndeksleme Yöntemi

Konumsal veriler genellikle çok boyutlu bir uzayda bir alan kaplarlar ve tek bir nokta ile ifade edilemezler (ülke sınırları, yollar vb.). Bu verilerin sorgulanması çoğunluk bir çerçeve içindeki coğrafi nesnelerin sorgulanması şeklindedir. (Guttman, 1984)

Veritabanı indeks yapısı tek boyutludur. Bu yüzden bilinen veritabanı indeksleme yöntemleri çok boyutlu konumsal sorgulamalar için uygulanabilir değildirler. (Guttman, 1984)

Çok boyutlu nokta verisi için birçok yapı geliştirilmiş ve araştırılmaktadır. Hücre yöntemi gibi yöntemler dinamik veriler için uygun değildirler çünkü hücre sınırları daha önceden belirlenmiş olmalıdır. Quad Tree ve K-D-B Tree yöntemleri sadece nokta yapısındaki nesneler için uygundur. (Guttman, 1984)

Konumsal veritabanı üstünde sadece nokta yapısına sahip objeler depolanmayacağı için R-Tree yöntemi araştırılmış ve çözümlenmiştir.

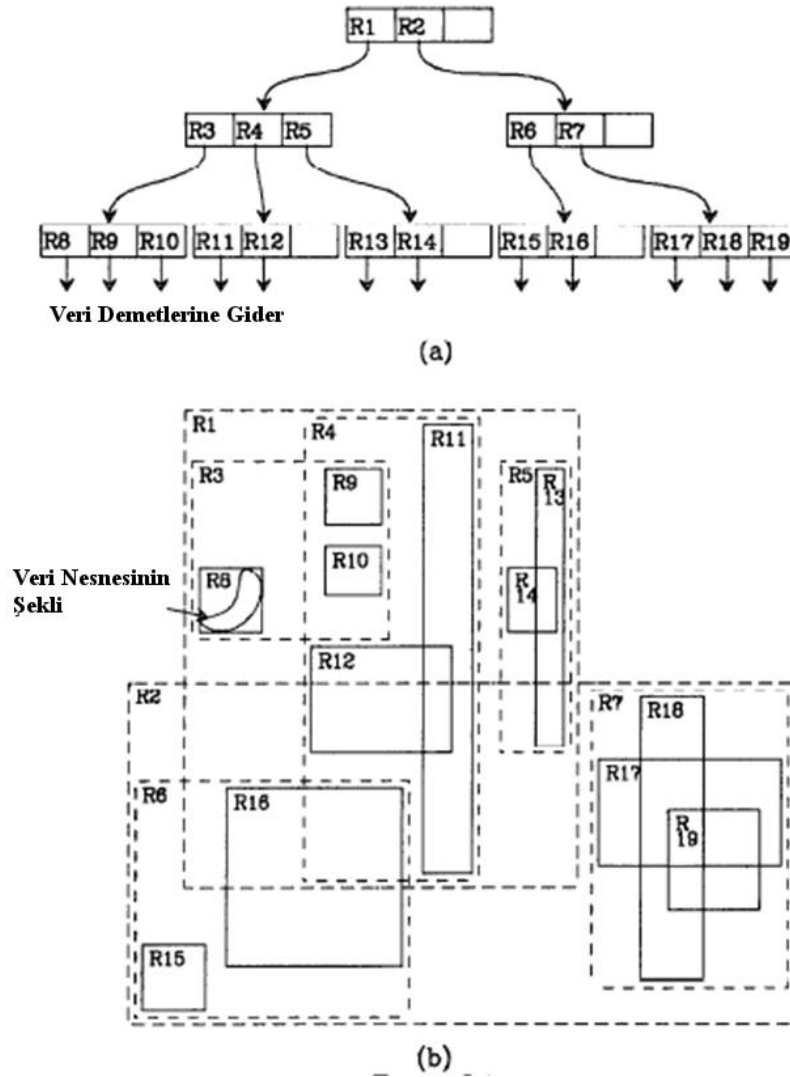
R-Tree yöntemi B-Tree yöntemiyle benzerlikler gösterir. Her düğüm noktası disk üzerindeki veriye ait referans içerir. Yöntemin temel özellikleri,

- “Root” dışındaki her dal düğümü m ve M indeks kayıtlarını içerir.
- Dal düğümü içindeki her indeks kaydı, objeyi içine alan I ile ifade edilen n boyutlu en küçük dikdörtgeni içerir.
- “Root” dışındaki dal düğümleri m ve M çocuklarıdır.

- d. Her düğümde mevcut bulunan I kendine bağlı dallarda bulunan girişlere ait I elemanlarını içeren dikdörtgendir.
- e. “Root” düğümünün en az iki çocuk bağlantısı (dalı) vardır.
- f. Her izin aynı seviyede gözükmemektedir.

3.1.1 R-Tree İndeks Kayıtları İçinde Arama Yapmak

R-Tree ağacı içinde aramaya root düğümünden başlanarak B-Tree yöntemiyle aynı prensipte arama yapılır. Tek fark yapılan aramadaki kıstas tek boyutlu olmadığı için Şekil 3.1’de gösterildiği gibi arama çerçevesi ile çakışan tüm dikdörtgenler aranır.



Şekil 3.1 (a) R-Tree Araması (b) R-Tree grafiksel gösterimi (Guttman, 1984)

3.1.2 R-Tree İndeks Kayıtlarına Ekleme Yapmak

Yeni eklenen çok boyutlu veri için R-Tree ağacının güncellenmesi B-Tree Ağacının güncellenmesi ile benzerdir. (Guttman, 1984)

Buna göre aşağıdaki işlem adımları uygulanır:

- Eklenen verinin indeks kaydı için doğru konumun bulunması
- Dal düğüm noktasına yeni indeks kaydının eklenmesi
- Dalların çoğaltılması durumunda üst bağlantıların güncellenmesi
- Eğer düğüm noktası bağlantılarının çoğaltılması durumunda yeni bir alt düğüm noktası gerekiyorsa ağacın uzatılması.

3.1.3 R-Tree İndeks Kayıtlarından Birinin Silinmesi İşlemi

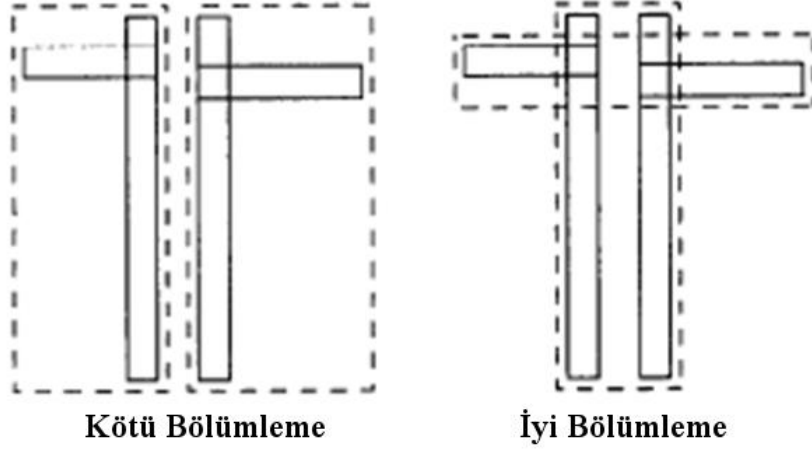
Veri yığını içinden silinen bir coğrafi obje beraberinde ilişkili indeks kaydının silinmesini gerektirir. Öncelikle indeks kaydı bulunur ve silinir. Bu işlem silinen indeks kaydının bağlı olduğu ve bunların bağlı olduğu düğüm noktasında I elemanlarının güncellenmesini gerektirir. (Guttman, 1984)

3.1.4 R-Tree İndeks Kayıtlarının Güncellenmesi

Veri yığını içindeki bir coğrafi nesnenin coğrafi özelliklerinin değişmesi I indeks kaydının değişmesi anlamına gelmektedir. Bu iş için öncelikle ilişkili indeks kaydı bulunur ve güncellenir. Bu işlem güncellenen indeks kaydının bağlı olduğu ve bunların bağlı olduğu düğüm noktasında I elemanlarının güncellenmesini gerektirir.

3.1.5 R-Tree Düğümlerinin Bölünmesi

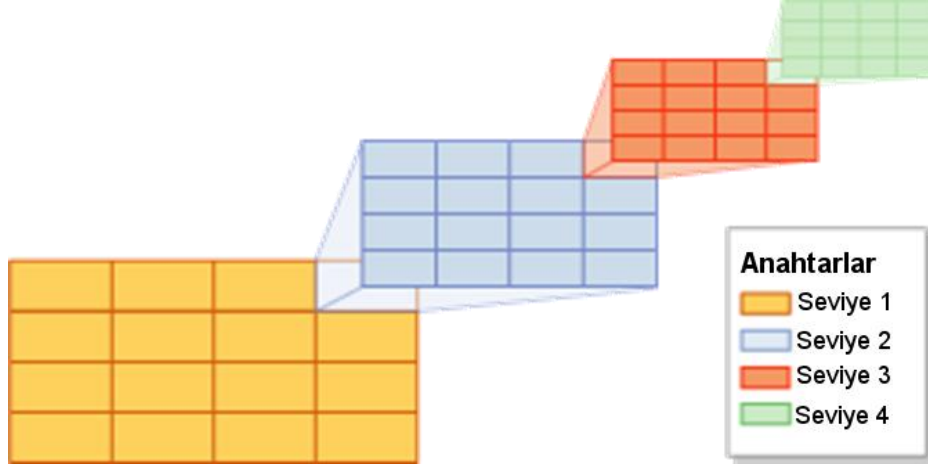
Veri yığını içine bir kayıt eklenmesi durumunda indeks düğüm noktaları dolu ise bölünmeleri gerekir. M düğüm noktasının $M+1$ düğüm noktası olacak şekilde genişletilmesi gerekmektedir. Buna göre ayırma işleminde oluşturulacak yeni düğüm yapısı için yeni I dikdörtgenleri oluşturulmalıdır. Bu dikdörtgenlerin oluşturulması sırasında en küçük alana sahip dikdörtgenler seçilmelidir. Şekil 3.2.'de kötü bölümlleme ile iyi bölümlleme grafik olarak gösterilmiştir.



Şekil 3.2 Yeni bölümlene sonrası oluşan yeni I elemanları (Guttman, 1984)

3.2 B-Tree Konumsal Veri İndeksleme Yöntemi

B-Tree konumsal veri indekslemesi coğrafi verilerin belirli seviyelerdeki gridler ile ayrıştırılması mantığına dayanır. SQL Server 2008 de B-Tree's yöntemini kullanmaktadır. SQL Server Şekil 3.3'de gösterilen dört seviyeli bir grid hiyerarşisi kullanır.



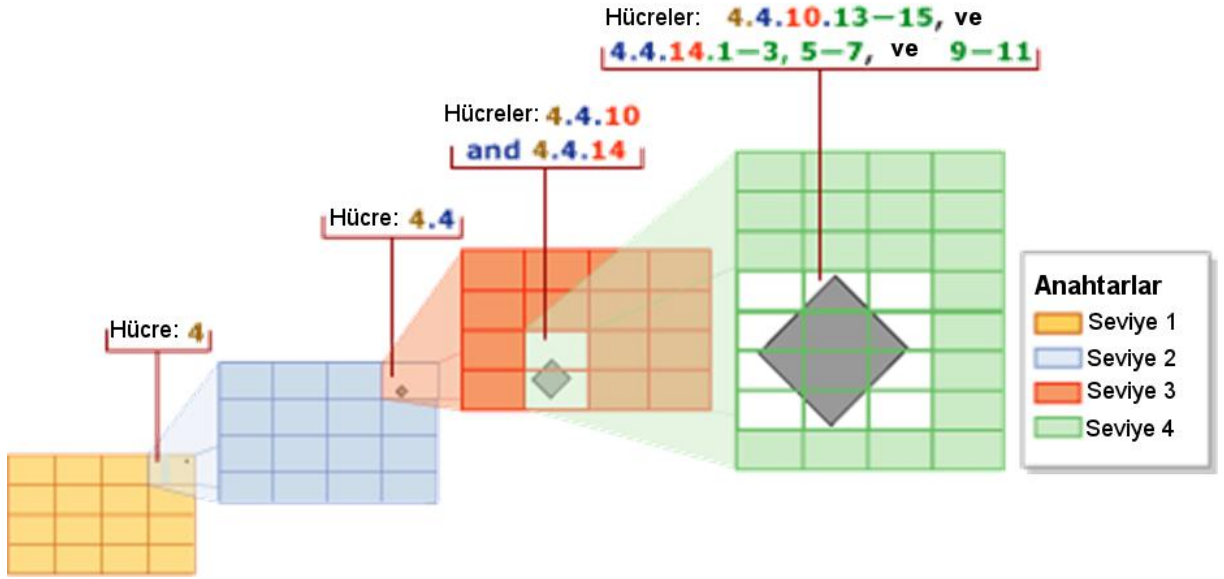
Şekil 3.3 Dört seviyeli grid hiyerarşisi (SQL Server 2008 Books Online)

Bu indeksleme yönteminde her bir hücre 4x4 (16 adet) alt hücre içerir. Bu değer grid yoğunluğu olarak ifade edilir. SQL Server 3 farklı yoğunluk seviyesi kullanır. Bir konumsal indeksin grid yoğunluğu CREATE_SPATIAL_INDEX komutu ile indeks oluşturulurken değiştirilebilir. SQL Server 2008'in desteklediği grid yoğunlukları ve farklılıkları Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1 Grid yoğunlukları (SQL Server 2008 Books Online)

Yoğunluk	Grid Yapılandırması	Hücre Sayısı
Düşük	4x4	16
Orta	8x8	64
Yüksek	16x16	256

Konumsal indeksleme ile veriye ulaşmak çok daha hızlanacaktır. Bu işlemin gerçekleşmesi mozaikleme ile mümkündür. Veriye ulaşmak istenildiğinde tablonun tamamı aranmadan önce indeks gridi içinde arama yapılır. Grid arama işleminin sonrasında bulunan gridlere dokunan nesneler getirilir. Bu işlem Şekil 3.4'de gösterilmektedir.



Şekil 3.4 Mozaikleme ile konumsal indeks kullanılarak yapılan arama (SQL Server 2008 Books Online)

3.3 B-Tree ve R-Tree Konumsal İndeksleme Yöntemlerinin Karşılaştırılması

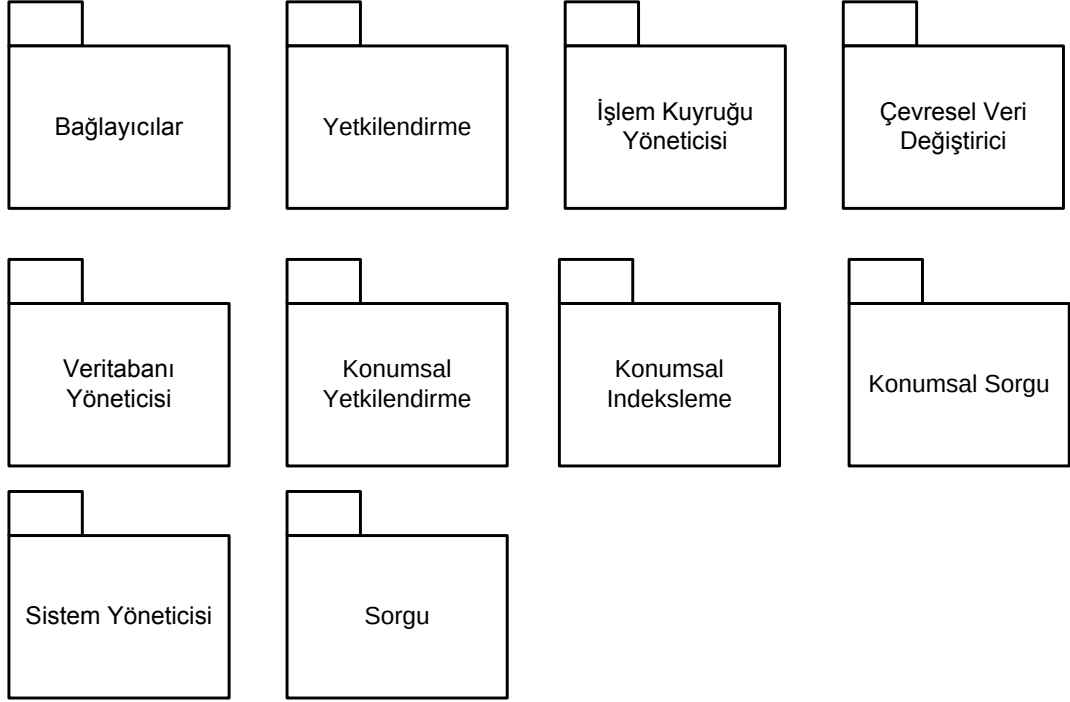
B-tree yöntemi ile R-tree yöntemi arasındaki temel ve en önemli farklılık B-tree yönteminde kullanılan hücre boyutlarının eşit, R-tree yönteminde kullanılan hücre boyutlarının farklı olmasıdır. Bu farklılık üstünde çalışılan veri türlerine göre çeşitli avantaj ve dezavantajlar oluşturmaktadır.

Buna ek olarak R-tree yönteminde indeks kayıtlarının oluşturulması indeks bölümlmesine karar verilmesinden dolayı B-tree yöntemine göre daha fazla zaman alır. Ayrıca uygulamada R-tree yönteminde oluşturulan indeksler fazla sayıdaki ekleme ve silme işlemlerinden sonra yeniden oluşturulmalıdır. Aksi halde yanlış veya verimsiz bölümlmeler ortaya çıkmaktadır. Bu da performans kaybına yol açmaktadır.

B-tree yöntemi ile oluşturulan indeks kayıtları R-tree kayıtlarından daha az yer kaplarlar. Buna karşılık R-tree yöntemi daha iyi bölümlleme yapar. Daha iyi bölümlleme veri okuma ve yazma işlemini çok kısaltır. Büyük verilerin tek parça olarak bölümlenmesi ise okuma, yazma ve değişiklik işlemlerinde büyük boyutlarda kayıtların kilitlenmesine sebep olur. (Lu, 2000)

4. SQL SERVER 2005 İÇİN SİSTEM TASARIMI

Yapılan araştırmalar, benzer sistem incelemeleri, teknolojik olabilirlik çalışmaları ve ihtiyaç analizleri sonucunda sistem modüler bir şekilde tasarlanmıştır. Tasarımın modülleri Şekil 4.1’de gösterilmiştir.



Şekil 4.1 Sistem modülleri

Öngörülen sistem modülleri;

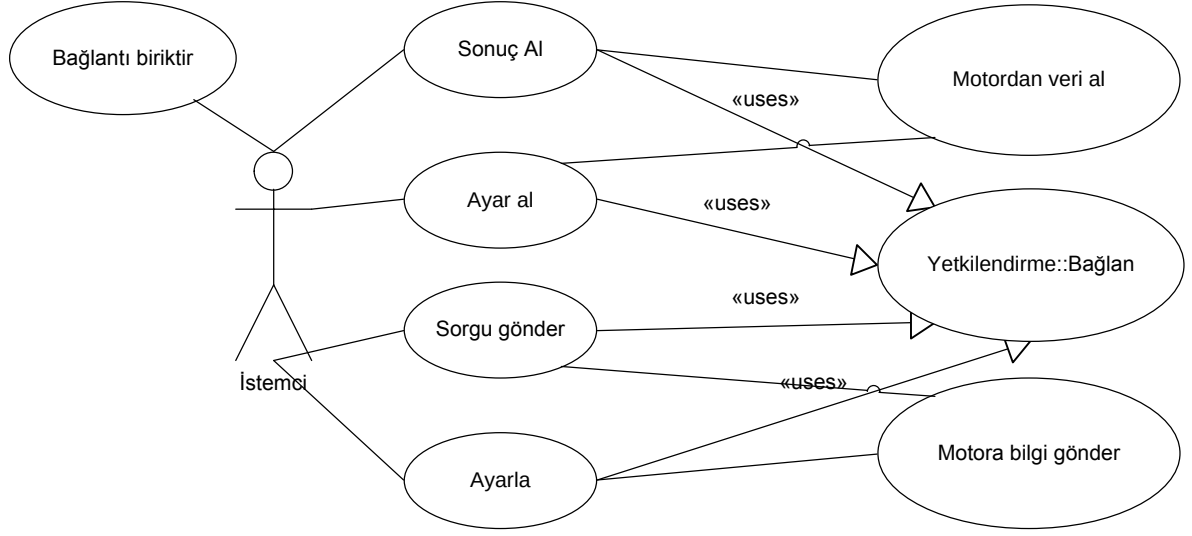
- Bağılayıcılar
- Veritabanı Yöneticisi
- Yetkilendirme
- İşlem Kuyruğu Yöneticisi
- Sorgu
- Çevresel Veri Değiştirici
- Sistem Yöneticisi
- Konumsal Yetkilendirme
- Konumsal İndeksleme
- Konumsal Sorgu

olarak belirlenmiştir.

4.1 BağılayıcılarModülü

Bağılayıcılar modülü sistemin dışarıya açılan tek arayüzü olarak tasarlanmıştır. Şekil 4.2’de bağılayıcılar modülü programların sisteme bağlanarak istedikleri işlemleri yapmalarına imkan

veren fonksiyon ve sınıflarla donatılmıştır.



Şekil 4.2 Bağlayıcılar modülü use case diyagramı

Bağlayıcılar modülü içinde kullanılan Aktör ve Use Case'ler:

İstemci: Sisteme bağlanan her türlü yazılımı temsil etmektedir.

Bağlantı biriktir: Sisteme bağlanan her kullanıcı için bellekte ayrılan alanları ve oluşturulan iş parçacıklarını yönetir. İstemci bağlantısını kestiği zaman kaynakları serbest bırakmak yerine bağlanacak olan sonraki istemcilerin depolar. Bu şekilde bağlantı kaynaklarının oluşturulma süresinden tasarruf edilerek performans artışı sağlanır. Bu use case işlemi gerçekleştirebilmesi için istemcinin yetkilerini Yetkilendirme Modülü'nün Bağlan Use Case'ine onaylatmak zorundadır.

Sonuç al: Sistemin istemci tarafından gönderilen Sorgusuna verdiği cevabı ilgili istemciye taşıyan birimdir. Dönen sorgunun ham sonucunu inceleyen bu birim, sonucu istenilen formata dönüştürür. Bu Use Case, işlemi gerçekleştirmek için istemcinin yetkilerini Yetkilendirme Modülü içindeki Login Use Case'ine onaylatmak zorundadır.

Ayar al: Sisteme bağlanan İstemcinin sistem ayarlarını okumasını sağlayan birimdir. Sorgu birimlerinden ayrı tutulmasının sebebi veri hırsızlığına sebebiyet verebilecek veritabanı ve işletim sistemi kaynaklarına erişim yetkilerine sahip hesap yetkilerine sahip olmasıdır.

Sorgu gönder: İstemci aktörünün Sisteme SQL sorgu dilinde komutlar göndermek için kullanacağı birimdir. Sorguyu analiz ederek sistemin hangi noktasına iletileceğini ve hangi kaynakları kullanacağını belirler. Tüm parametrelerle beraber sistemin içine gönderilmeye

hazır bir ileti paketi oluşturur. Bu Use Case işlemi gerçekleştirmek için istemcininyetkilerini YetkilendirmeModülü içindeki Login Use Case'ine onaylatmak zorundadır.

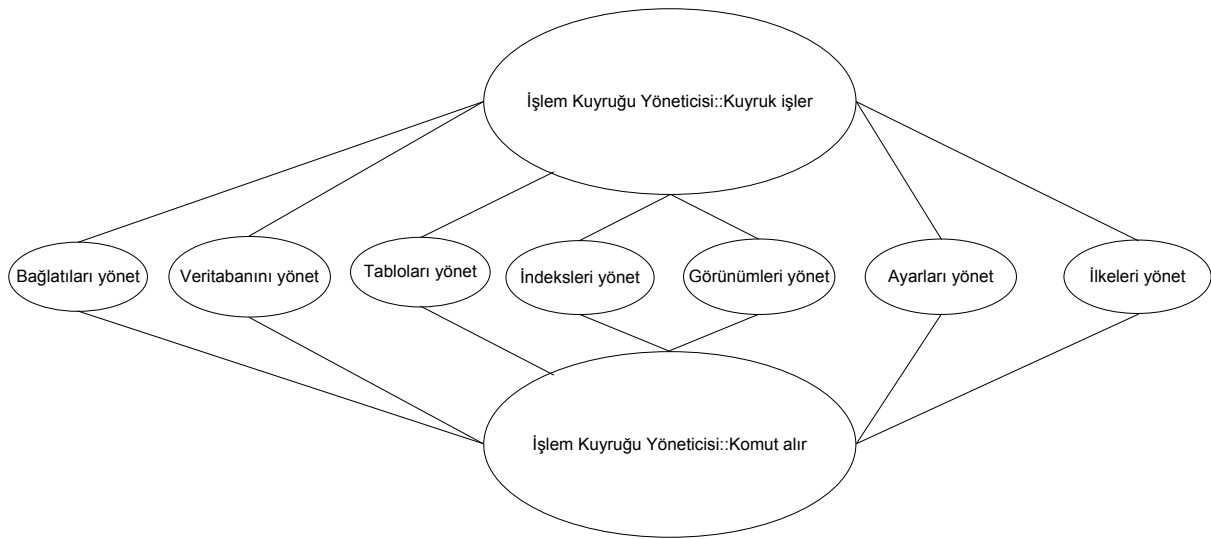
Ayarla: Sisteme bağlanan istemcininsistem ayarlarını değiştirmesini sağlayan bir bağlantı birimidir. Query birimlerinden ayrı tutulmasının sebebi güvenlik tehdidi oluşturabilecek olan veritabanı ve işletim sistemi kaynaklarına erişim yetkilerine sahip hesap yetkilerine sahip olmasıdır. Bu Use Case işlemi gerçekleştirmek için istemcininyetkilerini YetkilendirmeModülü içindeki Login Use Case'ine onaylatmak zorundadır.

Motordan veri al: Sistemin ilgili noktasından ulaşan paketleri bir posta merkezi gibi ayrıştırır ve ilgili istemciyeulaştırmak üzere gerekli birimlere teslim eder.

Motora veri gönder:İstemcidengelen istemleri ayrıştıran ve biçimleyen “Sorgu Gönder” ve “Ayarla” birimlerinden gelen veri paketlerini sisteme ulaştırmakla görevlidir.

4.2 Veritabanı Yönetim Modülü

Veritabanı Yönetim modülü sistemin kurallara uygun olarak veritabanı nesnelerini oluşturan ve yöneten modüldür. Şekil 4.3’de modüller arası oluşabilecek kullanım modülleri gösterilmektedir.Bu sebeple, konumsal bilgilerin depolandığı tabloların ve bunlarla ilişkili sistem tablolarının oluşturulmasını ve bu ilişkilerin kalıcı olarak devam etmesini garanti etmek için gerekli veritabanı ayarlarını yapmakla sorumlu olduğu için ayrı bir modül olarak tasarlanmıştır.



Şekil 4.3 Veritabanı yönetim modülü use case diyagramı

Veritabanı Yönetim modülü içinde kullanılan Use Case'ler:

Bağlantıları yönet: Sistem işlem kuyruğundan iletilen sistem kullanıcıları ile ilgili istemleri analiz ederek SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

Veritabanlarını yönet: Sistem işlem kuyruğundan iletilen veritabanları ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

Tabloları yönet: Sistem işlem kuyruğundan iletilen tablolar ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

İndeksleri yönet: Sistem işlem kuyruğundan iletilen indeksler ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

Görünümleri yönet: Sistem işlem kuyruğundan iletilen tablo görünümleri ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

Ayarları yönet: Sistem işlem kuyruğundan iletilen veritabanı ayarları ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

İlkeleri yönet: Sistem işlem kuyruğundan iletilen sistem ilkeleri ile ilgili istemleri analiz ederek gerekli SQL komutlarını oluşturur. Oluşturulan komutları SQL Server’da işletilmek üzere sistem işlem kuyruğuna iletir.

4.3 Yetkilendirme Modülü

Sistemde tanımlı kullanıcıların kendileri için önceden tanımlı kaynaklara erişimini denetleyen modüldür. Aynı zamanda birden fazla bilgisayarda çalışan sistemler için network üstünde akan sistem işletim bilgilerini şifrelemek ve şifrelerini çözmekle görevli modüldür.

Kullanıcıların sistem kaynaklarına erişimi üç farklı şekilde denetlenir. Bunlar yetkilendirme, güvenlik ilkeleri ve erişim ilkeleridir.

4.3.1 Yetkilendirme

Sistem içinde önceden tanımlanmış veritabanı ve sistem nesnelere erişim haklarını kontrol

ederek denetimi gerçekleştirir. Yetki denetimi işlemleri Şekil 4.4’de gösterildiği üzere izinleri ancak erişim talebinin erişim ve güvenlik ilkelerine de uygun olması durumunda verecektir.

Örneğin sistem içindeki bir tabloya erişmek isteyen bir kullanıcının daha önceden edinilmiş bir erişim hakkının olması gerekmektedir. Yetkilendirme mekanizması bu erişim hakkını kontrol eder.

4.3.2 Güvenlik İlkeleri

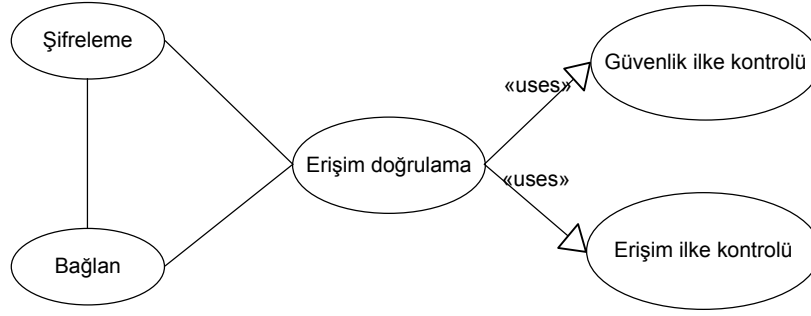
Güvenlik ilkeleri sistemin güvenliği ile ilgili ayarları içerir. Sistem’in her noktası güvenliğin sağlanabilmesi için bu ayarlara uymak zorundadır. Kullanıcılar da bu ilkelere uygun hareket etmek zorundadır.

Güvenlik ilkeleri kullanıcıların parola geçerlilik süreleri gibi ayarlardır. Güvenlik ilkelerinin çalışmasını açıklamak için bir örnek vermek gerekirse; sisteme bağlanma yetkisine sahip bir kullanıcının var olduğunu kabul edelim. Sistem parola geçerlilik süresinin bir ay olduğunu varsayalım. Kullanıcı parolasını en son 32 gün önce değiştirmiş olsun. Bu durumda kullanıcı sisteme bağlanmaya çalıştığında erişim kontrolü kullanıcının bağlanmasına izin verecektir ancak güvenlik ilkesi ile süzülen bu karar bozulacaktır. Bu şekilde ilkeler yetkilerin önüne geçerek sistemi ve içerdiği bilgileri koruyacaktır.

4.3.3 Erişim İlkeleri

Erişim ilkeleri Şekil 4.4’de gösterildiği üzere sistemin güvenliği için erişim parametreleri hakkındaki ayarları içerir. Sistem’in her noktası güvenliğin sağlanabilmesi için bu ayarlara uymak zorundadır. Kullanıcılar da bu ilkelere uygun hareket etmek zorundadır.

Erişim ilkelerine Sisteme bağlanma saatleri ve bağlantı yapmaya izin verilen IP adresleri örnek olarak verilebilirler. Saat 02:00 ila 03:00 saatleri arasında erişim ilkeleri ile kullanıma kapatılan sisteme bu saatler arasında bağlanmak isteyen bir kullanıcı yetkilendirme doğrulamasını ve güvenlik ilkelerini geçebilir, ancak bu iki bölüm tarafından olumlu yönde karar, erişim ilkeleri tarafından süzülerek bozulur. Bu şekilde erişim ilkeleri yetkilerin ve güvenlik ilkelerinin önüne geçerek sistemi ve içerdiği bilgileri koruyacaktır.



Şekil 4.4 Yetkilendirme modülü use case diyagramı

Yetkilendirme modülü içinde kullanılan Use Case'ler:

Bağlan: Kullanıcıların sisteme erişmek için yetkilerini denetleyen birimdir.

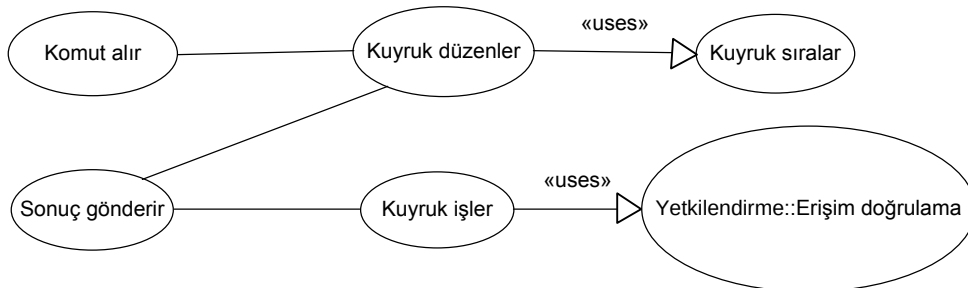
Güvenlik ilke kontrolü: Sisteme yapılan erişimlerin güvenlik ilkeleri ile uyumluluğunu denetleyen birimdir.

Erişim doğrulama: Kullanıcıların erişim yetkilerini denetleyen birimdir. Olumlu yönde karar verdiği erişim istemlerini “Güvenlik ilke kontrolü” ve “Erişim ilke kontrolü” birimlerine denetlettirir. Son kararı bundan sonra gerekli birime iletir.

Erişim ilke kontrolü: Sisteme yapılan erişimlerin erişim ilkelerine uyumluluğunu denetleyen birimdir.

4.4 İşlem Kuyruk Yöneticisi Modülü

İşlem Kuyruk Yöneticisi modülü Şekil 4.5’de de gösterildiği üzere sistemin tüm iş yükünü kullanıcı yetkilerine göre sıraya sokar, işletir ve sonuçlarını gerekli birime iletir. Sistemin omurgasını oluşturur.



Şekil 4.5 İşlem kuyruk yöneticisi modülü use case diyagramı

İşlem Kuyruk Yöneticisi modülü içinde kullanılan Use Case'ler:

Komut alır: Sistemin her noktasından gelen işlemlerin modüle giriş yaptığı birimdir.

Kuyruk düzenler: Kuyruğa eklenen işlemlerin performans artışını sağlamak üzere türlerine göre önceliklerini belirleyen birimdir.

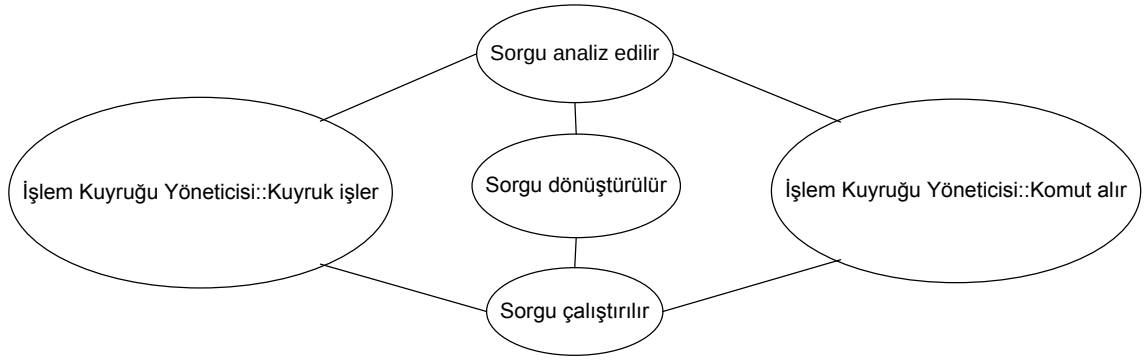
Kuyruk sıralar: Kuyruğa eklenmiş işlemlerin yetkisel önceliklerini belirleyerek kuyruğu düzenleyen birimdir.

Kuyruk işler: Kuyruktaki işlemleri işleten birimdir. İşletme öncesinde YetkilendirmeModülü içindeki Erişim Doğrulama birimine yetkileri denetlettirir.

Sonuç gönderir: İşlem sonuçlarını ilgili birime iletir.

4.5 SorguModülü

Sorgu modülü Şekil 4.6’da da gösterildiği üzere kullanıcı ve sistem tarafından kuyruğa eklenmiş olan SQL komutlarını veritabanı üstünde çalıştırır ve sonuçları geri döndürür.



Şekil 4.6 Sorgumodülü use case diyagramı

Sorgumodülü içinde kullanılan Use Case’ler:

Sorgu analiz edilir: Modüle sorgu komutlarının ulaştığı birimdir. Gelen komutu analiz ederek kurallara uygunluğunu denetler. Hatalı olan komutları geri çevirir.

Sorgu dönüştürülür: Sorgu komutu içindeki nesne isimleri ve özel komutları sistem içindeki adlarına dönüştüren birimdir. Ayrıca gerekli tür dönüşümlerini de gerçekleştirir.

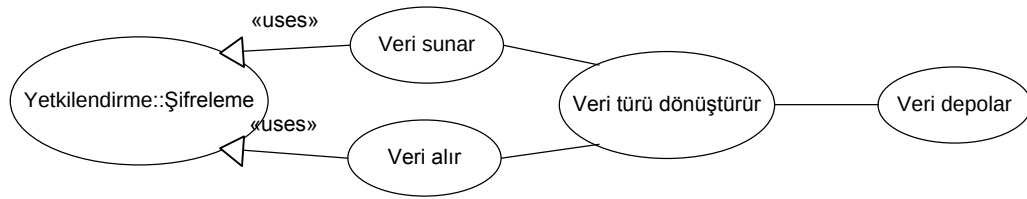
Sorgu çalıştırılır: Kontrol edilmiş ve dönüştürmüş komutları veritabanına göndererek çalıştırır. Dönen sonucu ilgili birime iletmek üzere işlem kuyruğuna gönderir.

4.6 Çevresel Veri Değiştirici Modülü

Çevresel Veri Değiştirici modülü Şekil 4.7’de de gösterildiği üzere sistem içinde global veri

değişimi işlevini yerine getirir. Sistem'e bağlanan her kullanıcı için ayrılmış özel bellek kaynakları ve sistem arasındaki veri geçişini sağlayan modüldür. Sistemle ilgili ayarların her defasında diskten okunması ciddi bir performans kaybına sebep olacaktır. Her kullanıcı için ayrılan özel bellekte kullanıcıya özel ayarlarla beraber sistem parametreleri de depolanır. Bu bilgilerden herhangi birinin değişmesi durumunda bağlı kullanıcıların anında yeni düzenlemelerden etkilenmesini sağlayacaktır.

Sistemin birden fazla bilgisayarda çalışması durumunda network iletişiminin şifrelenmesini destekleyecek şekilde tasarlanmıştır.



Şekil 4.7 Çevresel veri dönüştürücü modülü use case diyagramı

Çevresel Veri Dönüştürücü modülü içinde kullanılan Use Case'ler:

Veri sunar: Bilgileri ilgili birimlere ulaştıran birimdir.

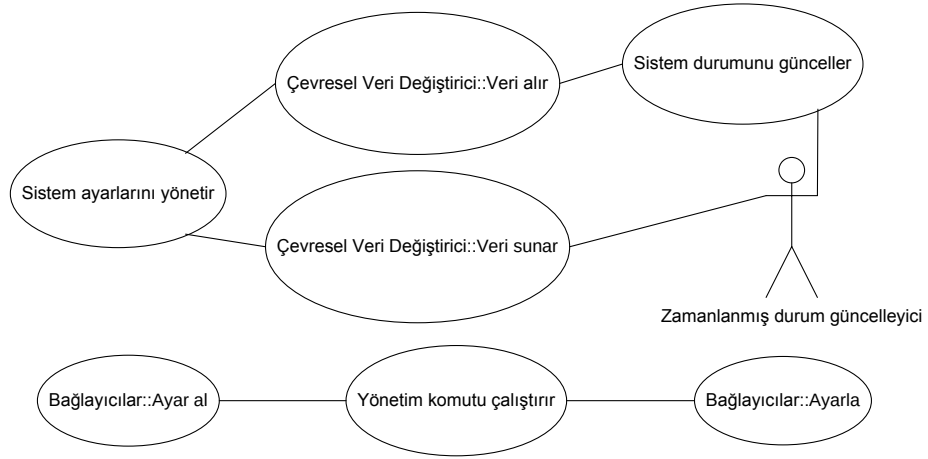
Veri alır: Modüle bilgilerin ulaştığı birimdir.

Veri türü dönüştürür: Gelen bilgilerin ortak bir veri dizisi içinde depolanabilmesi için tür dönüşümü gerçekleştiren birimdir.

Veri depolar: Gelen bilgileri depolayan veri dizisidir.

4.7 Sistem Yöneticisi Modülü

Sistem Yöneticisi modülü Şekil 4.8'de de gösterildiği üzere sistem ayarlarını depolar, istemcilere sağlar, sistemin çalışma durumunu iç hatalara karşı denetler ve yönetim komutlarını işler. İki ayrı erişim katmanından oluşmaktadır. İlki Sistem modüllerinin ayarları yönetmesini gerçekleştirir. İkincisi ise Bağlayıcılar modülü üstünden yapılan erişimleri gerçekleştirir.



Şekil 4.8 Sistem yöneticisi modülü use case diyagramı

Sistem Yöneticisi modülü içinde kullanılan Aktörler ve Use Case'ler:

Sistem ayarlarını yönetir: Sistem ayarlarını depolayan, sistem başlarken diskten okuyan ve uyumluluk kontrolünü yapan birimdir.

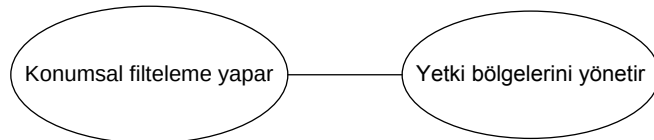
Sistem durumunu günceller: Sistem durumunu güncelleyen birimdir.

Yönetim komutu çalıştırır: Sistem ayarlarının istemciye ulaştırılmasını ve istemciden gelen ayar değişikliği komutlarını teslim alan birimdir.

Zamanlanmış durum güncelleyici: Sistem içinde yakalanamayan hatalardan dolayı oluşan aksaklıkları düzenli aralıklarla denetleyen aktördür.

4.8 Konumsal Yetkilendirme Modülü

Konumsal Yetkilendirme modülü Şekil 4.9'da da gösterildiği üzere sistem içinde konumsal yetkilendirmenin aktif hale getirildiği tabloları ve tablo görünümleri üstündeki okuma ve yazma filtrelendirmelerini yerine getirir. Ayrıca konumsal yetkilendirme ile ilgili sınır bilgilerinin yönetimi yerine getiren modüldür.



Şekil 4.9 Konumsal yetkilendirme modülü use case diyagramı

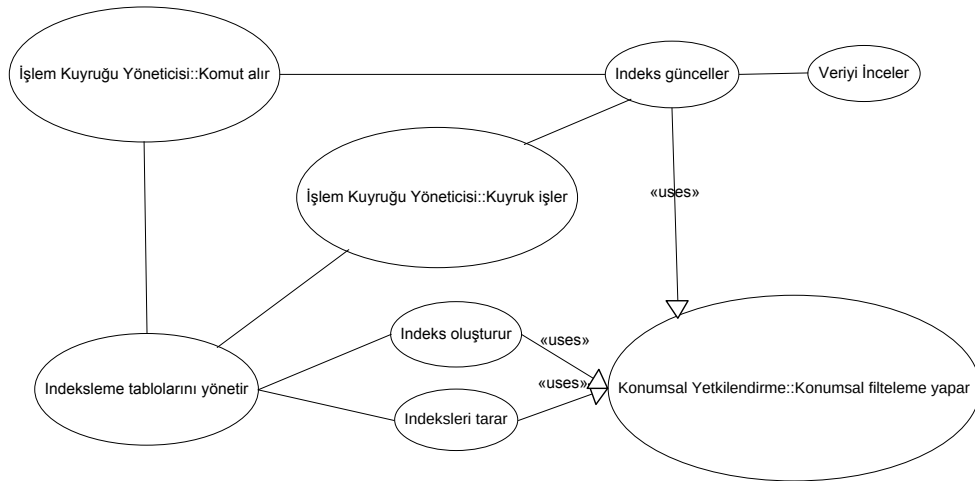
Konumsal Yetkilendirme modülü içinde kullanılan Use Case'ler:

Konumsal filtreleme yapar: Kullanıcının coğrafi nesnelere ve bunlarla ilişkili olan sözel bilgilere olan okuma, yazma, silme ve oluşturma yetkilerini coğrafi yetki sınırlarına göre süzen birimdir.

Yetki gölgelerini yönetir: Konumsal yetkilendirme için kullanılacak olan coğrafi sınırın yönetilmesi ve buna bağlı olarak gerekli konumsal ve sözel indeks düzenlemelerini yerine getiren birimdir.

4.9 Konumsal İndekslemeModülü

Konumsal İndekslememodülüŞekil 4.10’da da gösterildiği üzere coğrafi verilere ulaşma hızını artırmak için yapılması gerekli olan konumsal indeksleme işlemlerini yerine getirir. Bu modül oluşturulmuş olan tablolarla beraber oluşturulan sistem tabloları içinde indeks bilgilerini depolar ve yönetir.



Şekil 4.10Konumsal indeksleme modülü use case diyagramı

Konumsal İndekslememodülü içinde kullanılan Use Case’ler:

Veriyi inceler:İndeksleme için gerekli bilgileri toplayan birimdir.

İndeks günceller: Yapılan işleme göre indeks tablosuna yeni kayıt ekleyen, silen veya ilgili kaydı düzenleyen birimdir. Konumsal yetkilendirmenin aktifleştirildiği tablolarda indeksleme işlemlerinde yetkilendirme için oluşturulan ek indeksleri de düzenler.

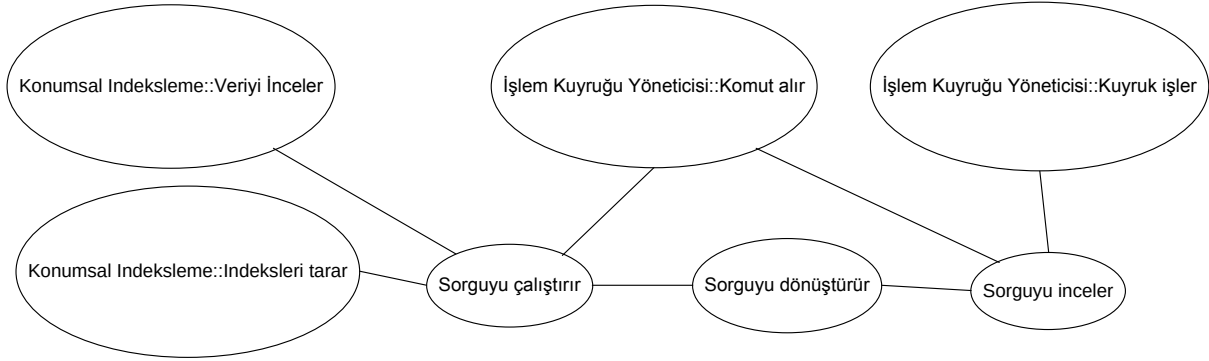
İndeks tablolarını yönetir: Konumsal indeksler için gerekli tabloları yöneten birimdir.

İndeksleri tarar: Bir sorgu sırasında ilgili nesnelere hızlı bir şekilde ulaşmak için indeks kayıtlarını tarayan birimdir.

İndeksleri oluşturur: Bazı durumlarda indeks tablosunun en baştan oluşturulması gerekir. Bu gibi durumlarda indeks tablosundaki tüm kayıtları silerek yeni kayıtları oluşturan birimdir.

4.10 Konumsal Sorgu Modülü

Konumsal Sorgu modülü Şekil 4.11’de de gösterildiği üzere kullanıcı tarafından gerçekleştirilecek olan konumsal sorgulamaları yerine getirir.



Şekil 4.11 Konumsal sorgu modülü use case diyagramı

Konumsal Sorgu modülü içinde kullanılan Use Case’ler:

Sorguyu çalıştırır: Kontrol edilmiş ve dönüştürmüş komutlar için gerekiyorsa konumsal indeks taraması yaparak komutu günceller ve veritabanına göndererek çalıştırır. Dönen sonucu ilgili birimi iletmek üzere işlem kuyruğuna gönderir. Bunun dışında konumsal indeks tablolarının güncel tutulması için ekleme, düzenleme ve silme durumlarında konumsal indeksleme modülünü bilgilendirir.

Sorguyu dönüştürür: Sorgu komutu içindeki nesne isimleri ve özel komutları sistem içindeki adlarına dönüştüren birimdir. Ayrıca gerekli veri türü dönüşümlerini de gerçekleştirir.

Sorguyu inceler: Modüle sorgu komutlarının ulaştığı birimdir. Gelen komutu analiz ederek kurallara uygunluğunu denetler. Hatalı olan komutları geri çevirir.

4.11 Tasarımın Geliştirilmesi Aşaması

Tez çalışmasında tasarım aşamasına 2006 yılında başlanmış ve 2007 yılının mayıs ayında tamamlanmıştır. Bu aşamdan sonra geliştirilme aşamasına başlanmıştır. Ancak 2007 yılının Kasım ayında Microsoft tarafından Microsoft SQL Server 2008’in “geometry” ve “geography” olmak üzere iki konumsal veri türünü destekleyeceği duyurulmuştur. Bu gelişmenin ardından veritabanı sunucusunun kendi bünyesinde barındırdığı konumsal veri

türleri ile çalışmanın hem daha performanslı hemde yeni teknolojiye uyumluluk açısından daha iyi olacağı görüldü ve tez izleme komitesi tarafından geliştirme işlemi durdurulmuştur.

Bu aşamadan sonra tez çalışmasına SQL Server 2008 üzerine odaklanarak devam edildi.

5. SQL SERVER 2008 YENİLİKLERİ

SQL Server 2008 halen Microsoft tarafından geliştirilmekte olan yeni bir veritabanıdır. İçerdiği teknolojilerin birçoğu SQL Server 2005'den miras kalmıştır. Bu özelliklerin dışında günümüz ihtiyaçları için tasarlanmış birçok yeni özellik içermektedir. SQL Server 2008 Books Online kaynağı incelenerek aşağıdaki bilgiler özetlenmiştir. (SQL Server 2008 Books Online)

Configuration Servers: Birden fazla sayıda SQL Server'ı yönetmek için geliştirilen bir özelliktir.

Policy-Based Management: Birden fazla sayıda SQL Server'ı olan, SQL Server'larla ilgili çok sayıda kişinin işlem yaptığı ortamlarda ilke tabanlı yönetim imkanı getiren bir özelliktir.

Resource Governor: SQL Server'ı paylaşan çeşitli kullanıcı istek gruplarının birbirlerinin performanslarını etkilediği durumlar için kullanılır. Hangi grubun ne kadar kaynak kullanabileceğini sınırlandırılabilir. Bu özellik ile daha iyi bir trafik yönetimi sağlanabilir.

Data Collector: SQL Server performansını ve sorunlarını bir veri ambarı yaklaşımı ile incelemeyi mümkün kılan özelliktir.

Plan Freezing: Bellekte yüklü planları daha detaylı yönetmeye imkan tanıyan yeni bir özelliktir.

Transparent Data Encryption: Uygulama katmanında hiç değişiklik yapmadan sadece veritabanı konfigürasyonu ile diskte verilerin şifreli olarak bulunmasını sağlayan yeni bir özelliktir.

Auditing All Actions: Veritabanı motorunu ve servislerinin anlık işlemlerini ve bununla ilgili birçok bilgiyi gözlemek için geliştirilen bir özelliktir.

FILESTREAM Data Type: Büyük dosyaların dosya sisteminde tutulması durumunda üzerlerinde yapılan işlemlerin kayıt altına alınması özelliğidir.

Spatial Data Support: Konumsal verileri tutan uygulamaları desteklemek için eklenen yeni veri türleridir.

Hot Add CPU: Daha iyi bir yüksek elverişlilik desteği sağlayan yeni bir özelliktir. Özellikle büyük uygulamalar için çok önemli bir yeniliktir.

Conflict Detection: Veri tekrarlarının tespit edilmesi için geliştirilen yeni bir özelliktir.

Change Data Capture: Birkaç prosedür çalıştırarak istenilen tabloların takibe alınmasını sağlayan yeni bir özelliktir. "Belirli bir tarih aralığında değişen kayıtlar hangileridir?" gibi soruların yanıtlarının kolaylıkla alınmasını sağlar.

MERGE statement: Verilerin daha akılcı birleştirilmesi için geliştiren yeni bir özelliktir.

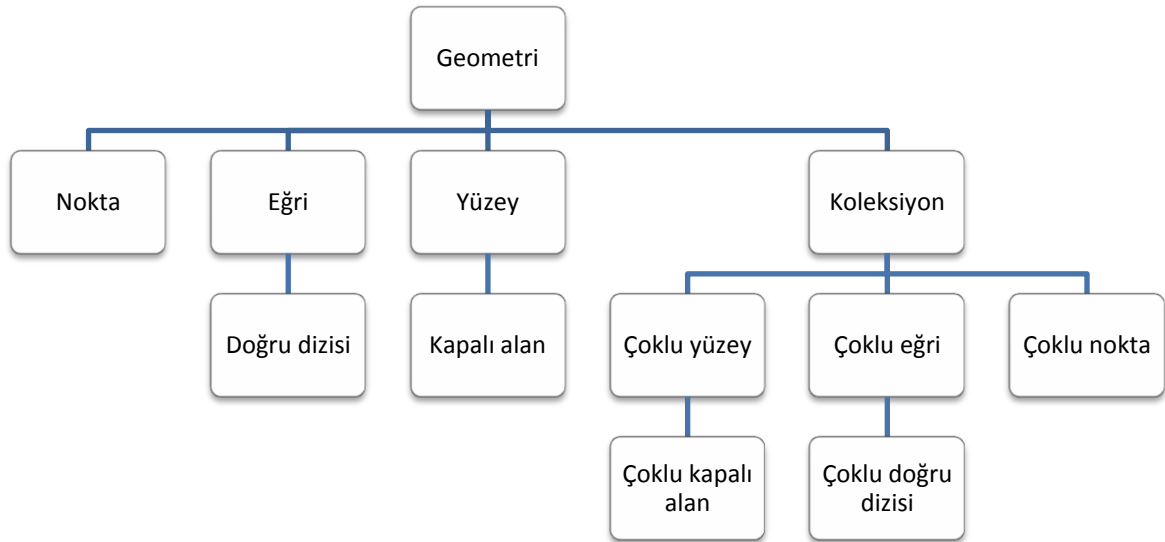
Report Designer: Teknik bilgisi olmayan kullanıcıların rapor geliştirmesine imkan veren yeni bir modüldür.

Analysis Services Wizards: Çok daha esnek küpler tasarlamak için Analysis Services içinde bulunan yeni sihirbazlardır.

5.1 SQL Server 2008 Konumsal Veri Türleri

SQL Server 2008 iki adet konumsal veri türü içerir. Bunlar geometry ve geography'dir. Geometry veri türü Open Geospatial Consortium (OGC) Simple Features for SQL Specification version 1.1.0. standartları doğrultusunda düzlemsel bir koordinat sistemi destekler. Geography veri türü ise elipsoidal (enlem, boylam) koordinatları destekler.

Her iki veri türü de on bir konumsal nesneyi destekler. Fakat bu nesnelerden sadece yedisi ile doğrudan çalışılabilir. Şekil 5.1 geometry ve geography veri türlerini ve konumsal nesneler arasındaki ilişkileri anlatır.



Şekil 5.1 Geometry ve geography veri türlerinin konumsal nesneler ile ilişkileri (SQL Server 2008 Books Online)

Her iki veri türü birbiri ile birçok yönden eşleşir. Ancak düzlemsel trigonometri ile küresel trigonometri arasındaki farklılıklar bu iki veri türü için de geçerlidir. Örneğin; iki nokta arasındaki uzunluk hesaplamasının düzlemsel ve küresel trigonometri açısından farklılığı

geometry ve geography veri türleri arasındaki farklılık ile eştir. Bu ve benzeri sistemsel farklılıklar dışında her iki veri türü birbirinin aynısıdır.

SQL Server 2008 konumsal veri türleri için birçok metot içerir. Bu metodlar OGC'nin standartlarına uygun geliştirilenleri ve Microsoft tarafından yapılan metodlar olarak ikiye ayrılırlar. OGC standartlarına ek olarak geliştirilen metodlar aşağıda listelenmiştir.

- AsGml (geometry Data Type)
- AsTextZM (geometry Data Type)
- BufferWithTolerance (geometry Data Type)
- InstanceOf (geometry Data Type)
- IsNull (geometry Data Type)
- M (geometry Data Type)
- MakeValid (geometry Data Type)
- Reduce (geometry Data Type)
- ToString (geometry Data Type)
- Z (geometry Data Type)

5.2 SQL Server 2008 Coğrafi Metodları

Sql Server 2008 coğrafi işlemler için birçok metot içerir. Bu metodlar aşağıdaki gibi gruplanabilir.

5.2.1 Coğrafi Veri Temsil Metodları

Bu komutlar ile T-SQL dili ile coğrafi nesneleri oluşturulabilir. Coğrafi nesneler bu şekilde SQL Server içinde sayılar veya metinler gibi ifade edilir.

Nokta		Çoklu Nokta	
Doğru dizisi		Çoklu Doğru dizisi	
Kapalı alan		Çoklu Kapalı alan	
Koleksiyon			

Şekil 5.2 Coğrafi nesne türleri (SQL Server 2008 Books Online)

Coğrafi veri temsil metodları geometry ve geography sınıflarının metodlarıdır. Bu metodlar üç kategoride incelenebilir. Bu kategoriler aşağıda listelenmiştir.

- **Well Known Binary (WKB):** Coğrafi nesnenin binary olarak ifade edilebilmesi için geliştirilmiş bir ifade sistemidir.
 - STPointFromWKB()
 - STMPPointFromWKB()
 - STLineFromWKB()
 - STLineFromWKB()
 - STMLineFromWKB()
 - STPolyFromWKB()
 - STMPolyFromWKB()
 - STGeomFromWKB()
 - STGeomCollFromWKB()

- **Well Known Text (WKT):** Coğrafi nesnelerin metin olarak ifade edilmesi için geliştirilmiş bir ifade sistemidir.
 - STPointFromText()
 - STMPPointFromText()
 - STLineFromText()
 - STMLineFromText()
 - STPolyFromText()
 - STMPolyFromText()
 - STGeomFromText()
 - STGeomCollFromText()

- **Geographic Markup Language (GML):** Coğrafi nesnelerin XML etiketleri ile ifade edildiği bir sistemdir.
 - GeomFromGml()

5.2.1.1 İyi Bilinen İkili Değer Yöntemi (Well Known Binary)

Well Known Binary (WKB) coğrafi nesnelerin ikili sayı sistemi şeklinde kodlanmış olarak ifade edilmesi için OGC tarafından geliştirilmiş bir sistemdir. WKB gösterimi WKT gösterimi ile karşılaştırıldığında WKT kadar anlaşılır değildir. Buna rağmen GML ve WKT gösterimlerine göre avantajları da vardır. Bunlar;

- WKB benzeri olan GML ve WKT yöntemlerinden daha hızlıdır. Çünkü veritabanı motoru WKT ve GML verilerini anlayabilmek için öncelikle metin formatındaki bu veriyi ayrıştırmalı ve gerekli dönüşümleri yapmalıdır. Oysa WKB ile ifade edilmiş bir coğrafi veri için ayrıştırma ve dönüşüm işlemlerini yapmamaktadır.
- WKB yönteminde ondalıklı sayılar binary olarak ifade edildiği için yuvarlama hataları WKT ve GML yöntemlerine göre çok daha düşüktür.
- WKB çoğu zaman WKT ve GML'den boyut olarak daha küçüktür. Bu sayede daha hızlı aktarılabilir. Disk ve hafıza üstünde daha hızlı erişilebilirler. Bu çok fazla verinin olduğu sistemlerde çok önemlidir.

WKB yönteminin dezavantajı ise insanlar tarafından kolaylıkla anlaşılamamasıdır. Bundan dolayı hata yakalama ve giderme işlemleri çok zordur ve çok vakit alır. Tüm bu sebeplerden dolayı Well Known Binary yöntemi daha çok makineler/yazılımlar/sistemler arasındaki iletişimde tercih edilirler.

Her binary veri formatı gibi başlık ve veri kısımlarından oluşmaktadır. Başlık kısmında veri alanındaki byte'ların hangi coğrafi nesneyi temsil ettiği bilgisi vardır. Başlık kısmı coğrafi alandaki verinin cinsine göre daha fazla bilgi içerebilir. Veri kısmı 16 byte'lık paketlerden oluşur ve her paketin ilk 8 byte'ı x koordinatını son 8 byte'ı y koordinatını içerir.

[Sıralama Sistemi] [Nesne Türü] [X] [Y]

Veri kısmındaki byte'lar Big Endian veya Little Endian olmak üzere iki şekilde sıralanabilir. Aşağıdaki örnek WKB bir noktayı temsil etmektedir ve Big Endian olarak sıralanmıştır. Her iki karakter 16'lık sayı sisteminde 00 ila FF (255) arasında bir değeri ifade eder.

0x0000000001401C0000000000000040300E000A000000

Big Endian sıralama sisteminde ilk byte 00'dır ve byte dizilimi soldan sağa şeklindedir.

0x00 00000001401C000000000000004030000000000000

Little Endian sıralama sistemindeki ilk byte 01'dir ve byte dizilimi sağdan sola şeklindedir.

0x01010000000000000000000001C400000000000003040

Well Known Binary Sisteminde Coğrafi Nesnelerin İfade Biçimleri

Nokta'nın WKB olarak ifadesi;

[Sıralama Sistemi] [0x00000001] [X] [Y]

Polyline'nın WKB olarak ifadesi ;

```
[Sıralama Sistemi] [0x00000002] [Nokta Sayısı (4byte)] [X1]
[Y1] [X2] [Y2] ... [Xn] [Yn]
```

Kapalı alanın WKB olarak ifadesinde halka olarak adlandırılan nesneler kullanılır. Buna göre kapalı alanlar halkalardan oluşmaktadır ve halkalarda noktalardan oluşmaktadır. Kapalı alanın WKB olarak ifadesi;

```
[Sıralama Sistemi] [0x00000003] [Halka Sayısı (4byte)]
[Halka1] [Halka2] ... [HalkaN]
```

Her halka aşağıdaki şekilde ifade edilir;

```
[Nokta Sayısı] [X1] [Y1] [X2] [Y2] ... [Xn] [Yn]
```

Çoklu geometrik nesnelerin WKB olarak ifadesi basitçe aşağıdaki gibidir;

```
[Sıralama Sistemi] [Nesne Türü] [Geometrik Nesne Sayısı
(4byte)] [Nesne1] [Nesne2] ... [NesneN]
```

Çoklu coğrafi nesnelerin WKB olarak ifadesinde nesne türleri;

- MultiPoint için 4,
- MultiLineString için 5,
- MultiPolygon için 6,
- Geometry Collection için 7 olarak belirlenmiştir.

Her nesne kendi içinde bir WKB formatı şeklindedir. Ve kendine özel ek parametreleri de içerir.

5.2.1.2 İyi Bilinen Metin Yöntemi (Well Known Text)

İyi Bilinen Metin yöntemi OGC tarafından geliştirilen coğrafi nesnelerin insanların okuyabileceği karakterler ile metinsel ifadesidir. Çoğunlukla insan ile makine arasındaki iletişim için kullanılırlar. Neredeyse tüm sistemler verileri binary olarak depolarlar. Binary veri insanlar tarafından zorlukla anlaşıldığı için verilerin grafiksel gösterimi veya metinsel gösterimi gerekmektedir. Bunun için aşağıdaki kodlama sistemi geliştirilmiştir.

- POINT(x y) nokta tanımlamak için kullanılır.
- LINESTRING(x1 y1, x2 y2, ... , xn yn) Polyline tanımlamak için kullanılır.

- POLYGON($x_1 y_1, x_1 y_1, x_2 y_2, \dots, x_n y_n, x_1 y_1$) Kapalı alan tanımlamak için kullanılır
- MULTIPOINT($x_1 y_1, x_2 y_2, \dots, x_n y_n$) aynı anda birden çok nokta tanımlamak için kullanılır.
- MULTILINESTRING($(x_1 y_1, x_2 y_2, \dots, x_n y_n), (x_1 y_1, x_2 y_2, \dots, x_n y_n)$) aynı anda birden çok polyline tanımlamak için kullanılır.
- MULTIPOLYGON($(x_1 y_1, x_1 y_1, x_2 y_2, \dots, x_n y_n, x_1 y_1), (x_1 y_1, x_1 y_1, x_2 y_2, \dots, x_n y_n, x_1 y_1)$) aynı anda birden çok kapalı alan tanımlama için kullanılır.
- GEOMETRYCOLLECTION (POINT($x y$), LINESTRING($x_1 y_1, x_2 y_2$), ...) farklı coğrafi nesneleri aynı anda bir kolleksiyon olarak tanımlamak için kullanılır.
- POINT ZM ($x y z m$) 4 boyutlu bir nokta tanımlama için kullanılır.
- POINT M ($x y z$) üç boyutlu nokta tanımlamak için kullanılır.

Türkiye ülke sınırlarının İyi Bilinen Metin yöntemi ile ifade edilmesi için MULTIPOLYGON komutu kullanılır. İyi Bilinen Metin olarak Türkiye ülke sınırları şu şekilde ifade edilir:

```
MULTIPOLYGON(((26.7638912200928 40.6599884033203 0 0,26.2115936279297 40.3221473693848 0
0,26.1644058227539 40.0519714355469 0 0,27.5053424835205 40.9813079833984 0 0,29.036247253418
41.0545768737793 0 0,29.0888824462891 41.2462425231934 0 0,28.0905494689941 41.6313858032227 0
0,28.013053894043 41.9822158813477 0 0,27.569580078125 41.9092636108398 0 0,27.0702705383301
42.0899887084961 0 0,26.3610954284668 41.711051940918 0 0,26.6357593536377 41.3647155761719 0
0,26.3249969482422 41.2347106933594 0 0,26.3604125976563 40.9538803100586 0 0,26.0447196960449
40.7358245849609 0 0,26.1186084747314 40.5951271057129 0 0,26.7638912200928 40.6599884033203 0
0)),((44.7873382568359 37.1497116088867 0 0,44.6177673339844 37.7179756164551 0 0,44.2239685058594
37.8991508483887 0 0,44.4825172424316 38.3413009643555 0 0,44.3052597045898 38.4005355834961 0
0,44.0349540710449 39.3774452209473 0 0,44.4011001586914 39.4165153503418 0 0,44.6084632873535
39.7791557312012 0 0,44.8130416870117 39.6308135986328 0 0,44.8163757324219 39.6458206176758 0
0,44.7788619995117 39.7063827514648 0 0,44.3516616821289 40.0222206115723 0 0,43.6683959960938
40.1031799316406 0 0,43.5830535888672 40.4511108398438 0 0,43.7504081726074 40.7449989318848 0
0,43.4607696533203 41.1129608154297 0 0,42.8310203552246 41.5824165344238 0 0,42.4722137451172
41.4333267211914 0 0,41.5315589904785 41.5238761901855 0 0,40.1499938964844 40.9202728271484 0
0,39.4142265319824 41.1067314147949 0 0,38.3558235168457 40.910270690918 0 0,36.8099975585938
41.3555526733398 0 0,36.4315185546875 41.2420768737793 0 0,36.0561027526855 41.6887435913086 0
0,35.2866516113281 41.7130508422852 0 0,35.0063858032227 42.0860977172852 0 0,34.7155456542969
41.9424896240234 0 0,33.3386001586914 42.0198516845703 0 0,32.2774963378906 41.7194366455078 0
0,31.2334690093994 41.089298248291 0 0,29.1599998474121 41.2245750427246 0 0,29.1294403076172
40.9144439697266 0 0,29.9339542388916 40.7222099304199 0 0,28.7767295837402 40.5270767211914 0
0,29.0557613372803 40.3667984008789 0 0,27.9531936645508 40.35693359375 0 0,28.020133972168
40.4877052307129 0 0,27.7548580169678 40.5297203063965 0 0,27.8778419494629 40.3758583068848 0
0,27.511661529541 40.3055534362793 0 0,27.1186065673828 40.452766418457 0 0,26.7043018341064
40.3788795471191 0 0,26.1574974060059 39.9466590881348 0 0,26.1099948883057 39.4576301574707 0
0,26.9518013000488 39.5508270263672 0 0,26.6447219848633 39.263053894043 0 0,27.0629806518555
38.8731842041016 0 0,26.7337455749512 38.6419372558594 0 0,27.1564521789551 38.4528045654297 0
0,26.691801071167 38.3108558654785 0 0,26.3983306884766 38.6669387817383 0 0,26.5098571777344
38.425479888916 0 0,26.2758293151855 38.2644348144531 0 0,27.2683296203613 37.9535980224609 0
0,27.1949253082275 37.3523483276367 0 0,27.5952758789063 37.2324905395508 0 0,27.2549991607666
```

```

36.9649963378906 0 0,28.3247871398926 37.0381851196289 0 0,27.374719619751 36.684024810791 0
0,28.1201362609863 36.7970771789551 0 0,27.98388671875 36.5527725219727 0 0,28.4563865661621
36.8808288574219 0 0,29.0524978637695 36.6811065673828 0 0,29.1480503082275 36.3483276367188 0
0,29.6772155761719 36.1183319091797 0 0,30.2086086273193 36.3038749694824 0 0,30.404857635498
36.2049217224121 0 0,30.6165256500244 36.8436050415039 0 0,31.0466613769531 36.8491516113281 0
0,32.0647125244141 36.5163879394531 0 0,32.7716598510742 36.0288848876953 0 0,33.9886016845703
36.2777709960938 0 0,34.6594314575195 36.8052749633789 0 0,35.3470726013184 36.5449905395508 0
0,36.0112037658691 36.9231910705566 0 0,36.2172813415527 36.6547813415527 0 0,35.7855453491211
36.3148460388184 0 0,35.9224395751953 35.9269943237305 0 0,36.1684684753418 35.8197174072266 0
0,36.392219543457 36.2133255004883 0 0,36.6831169128418 36.2327346801758 0 0,36.5495758056641
36.4877700805664 0 0,36.6599426269531 36.8337097167969 0 0,37.3738136291504 36.6553382873535 0
0,38.2433967590332 36.9138793945313 0 0,39.2215232849121 36.665340423584 0 0,40.7708206176758
37.118049621582 0 0,41.5481872558594 37.0852737426758 0 0,42.1808319091797 37.2905426025391 0
0,42.3556137084961 37.1069259643555 0 0,42.7868003845215 37.383674621582 0 0,44.1188812255859
37.3156852722168 0 0,44.3191604614258 36.9712371826172 0 0,44.7873382568359 37.1497116088867 0
0)),((27.6016616821289 40.5719375610352 0 0,27.6093006134033 40.5727691650391 0 0,27.7069435119629
40.6499938964844 0 0,27.5459671020508 40.6549911499023 0 0,27.6016616821289 40.5719375610352 0
0)),((25.7316665649414 40.0930404663086 0 0,25.9580535888672 40.1211013793945 0 0,25.9405498504639
40.239990234375 0 0,25.6888885498047 40.1649856567383 0 0,25.7316665649414 40.0930404663086 0 0)))

```

5.2.1.3 Coğrafi İşaretleme Dili (Geographic Markup Language)

GML coğrafi nesnelerin XML etiketleri ile ifade edilebilmesi için OGC tarafından geliştirilmiş bir kodlama sistemidir. GML doküman yapısında geliştirilmiştir ve çok esnektir. Her türlü coğrafi veri kolaylıkla GML olarak temsil edilebilir.

```

<Point xmlns="http://www.opengis.net/gml">
<pos>10 30</pos>
</Point>

```

Noktanın GML olarak ifade edilmesi:

```

<Point xmlns="http://www.opengis.net/gml">
<pos>10 30</pos>
</Point>

```

PoliLine'nın GML olarak ifade edilmesi:

```

<LineString xmlns="http://www.opengis.net/gml">
<posList>10 30 22 56</posList>
</LineString >

```

Kapalı alanların GML olarak ifade edilmesi:

```

<Polygon xmlns="http://www.opengis.net/gml">
<exterior>
<LinearRing>
<posList>0 0 100 100 0 100 0 0</posList>

```

```

</LinearRing>
</exterior>
<interior>
<LinearRing>
<posList>10 10 20 10 20 20 10 20 10 10</posList>
</LinearRing>
</interior>
<interior>
<LinearRing>
<posList>75 10 80 10 80 20 75 20 75 10</posList>
</LinearRing>
</interior>
</Polygon>

```

Çoklu noktaların GML olarak ifadesi:

```

<MultiPoint xmlns="http://www.opengis.net/gml">
<pointMembers>
<Point>
<pos>2 3</pos>
</Point>
<Point>
<pos>4 10</pos>
</Point>
</pointMembers>
</MultiPoint>

```

Çoklu polilinelerin GML olarak ifadesi:

```

<MultiCurve xmlns="http://www.opengis.net/gml">
<curveMembers>
<LineString>
<posList>2 3 4 10</posList>
</LineString >
<LineString>

```

```
<posList>4 10 15 40</posList>
```

```
</LineString>
```

```
</curveMembers>
```

```
</MultiCurve >
```

Çoklu kapalı alanların GML olarak ifade edilmesi:

```
<MultiSurface xmlns= "http://www.opengis.net/gml ">
```

```
<surfaceMembers>
```

```
<Polygon>
```

```
<exterior>
```

```
<LinearRing>
```

```
<posList>2 3 5 3 6 8 2 7 2 3</posList>
```

```
</LinearRing>
```

```
</exterior>
```

```
</Polygon>
```

```
<Polygon>
```

```
<exterior>
```

```
<LinearRing>
```

```
<posList>10 20 20 20 20 30 10 30 10 20</posList>
```

```
</LinearRing>
```

```
</exterior>
```

```
</Polygon>
```

```
</surfaceMembers>
```

```
</MultiCurve >
```

Çoklu geometrik nesneleri GML olarak ifade edilmesi:

```
<MultiSurface xmlns= "http://www.opengis.net/gml">
```

```
<geometryMembers>
```

```
<Point>
```

```
<pos>15 10</pos>
```

```
</Point>
```

```
<LineString>
```

```
<posList>4 10 2 3</posList>
```

```
</LineString >
```

```
</geometryMembers>
```

```
</MultiSurface>
```

5.2.2 Sayısal geometri komutları

Bu komutlar ile sayısalgeometri (Computational geometry) problemleri çözümlenir. SQL Server 2008 tarafından desteklenen komutlar şunlardır;

STArea: Coğrafi nesnenin alanını hesaplar.

STASBinary: Coğrafi nesnenin Binary formattaki ifadesini geri döndürür.

STAsText: Coğrafi nesnenin Text formattaki ifadesini geri döndürür.

STBoundary: Coğrafi nesnenin sınırını geri döndürür.

STBuffer: Coğrafi nesneden belirtilen mesafe kadar daha büyütülmüş bir coğrafi nesne geri döndürür.

STCentroid: Kapalı alanınveya alanların merkez noktasını geri döndürür.

STContains: Bir coğrafi nesnenin diğer bir coğrafi nesneyi tamamıyla kapsayıp kapsamadığını test eder. Geri dönen sonuç 1 ise nesne tamamıyla diğer nesnenin içindedir.

STConvexHull: Bir coğrafi nesneyi veya coğrafi nesneleri çevreleyen en küçük kapalı alanı geri döndürür.

STCrosses: Bir coğrafi nesne ile diğer bir coğrafi nesnenin kesişip kesişmediğini test eder. Geri dönen sonuç 1 ise kesişim vardır.

STDifference: İki coğrafi nesnenin farkını geri döndürür.

STDimension: Coğrafi nesnenin boyutunu geri döndürür. Buna göre 0: nokta, 1: poliline, 2: kapalı alandır

STDisjoint: Coğrafi nesnelerin bir birleri ile temasını, kesişimini ve örtüşmesini test eder. Eğer geri dönen sonuç 1 ise herhangi bir kesişim, temas veya örtüşme söz konusu değildir.

STDistance: İki coğrafi nesne arasındaki en kısa uzaklılığı geri döndürür.

STEndPoint: Coğrafi nesnenin son noktasını geri döndürür.

STEnvelope: Bir coğrafi nesneyi çevreleyen eksnlere paralel en küçük dikdörtgeni geri döndürür.

STEquals: İki coğrafi nesnenin bir birine eşit olup olmadığını test eder. Eğer geri dönen

sonuç 1 ise nesneler birbirlerine eşittir.

STExteriorRing: Coğrafi objeyi çevreleyen kapalı alanı geri döndürür. Sadece Geometry nesne türü tarafından desteklenir.

STGeometryN: Bir geometri koleksiyonu içindeki belirli bir sıradaki nesneyi geri döndürür.

STGeometryType: Coğrafi nesnenin OGC tarafından verilen tür adını geri döndürür.

STInteriorRingN: Coğrafi nesne içindeki deliğin sınırlarını kapalı alan olarak geri döndürür.

STIntersection: İki coğrafi nesnenin kesişip kesişmediğini test eder ve kesişim noktasını/noktalarını geri döndürür. Geri dönen sonuç null ise kesişim yoktur.

STIntersects: İki coğrafi nesnenin kesişip kesişmediğini test eder. Geri dönen sonuç 1 ise kesişim vardır.

STIsClosed: Coğrafi objenin bir kapalı alan oluşturup oluşturmadığını kontrol eder. Geri dönen sonuç 1 ise coğrafi nesne kapalı alandır.

STIsEmpty: Coğrafi nesnenin boş olup olmadığını test eder. Geri dönen sonuç bir ise nesne boştur.

STIsRing: Coğrafi nesnenin aynı anda basit ve kapalı alan olup olmadığını test eder. Geri dönen sonuç bir ise hem basit bir nesnedir hem de kapalı bir alandır.

STIsSimple: Coğrafi nesnenin OGC standartlarında tanımlanan basit bir nesne olup olmadığını test eder. Geri dönen sonuç bir ise nesne basit bir nesnedir.

STIsValid: SQL server içinde noktaların kombinasyonu ile birçok şekil oluşturulabilmekte ve depolanabilmektedir. Ancak SQL server bu nesneleri yorumlarken hata yapabilir. Örneğin bir kare kendi üstüne doğru bükülmüş şekilde bir fiyonk gibi depolanabilir. Bu metot ile bu tür biçimsel anormalliklerin var olup olmadığı sorgulanabilir. Bu metot sadece geometry nesnesi tarafından desteklenmektedir.

STLength: Coğrafi nesnenin toplam uzunluğunu geri döndürür.

STNumGeometries: Coğrafi nesnenin içerdiği nesne sayısını geri döndürür.

STNumInteriorRing: Coğrafi nesnenin içerdiğini delik sayısını geri döndürür.

STNumPoints: Coğrafi nesnenin içerdiği nokta sayısını geri döndürür.

STOverlaps: İki coğrafi nesnenin örtüşüp örtüşmediğini test eder. Geri dönen sonuç 1 ise

coğrafi nesneler bir birleri ile örtüşmektedir.

STPointN: Coğrafi nesneyi oluşturan noktalardan belirli bir sıradakini geri döndürür.

STPointOnSurface: Coğrafi nesne üstünde bir nokta geri döndürür.

STRelate: İki coğrafi nesnenin kesişim matrisi ile tanımlanan bir ilişkiye sahip olup olmadığını test eder. Geri dönen sonuç 1 ise iki nesne ilişkilidir.

STSrid: Coğrafi nesnenin tanımlayıcısını geri döndürür.

STStartPoint: Coğrafi nesnenin ilk noktasını geri döndürür.

STSymDifference: İki coğrafi nesnenin kesişmediği alanları birleştirerek geri döndürür. STIntersection metodunun tersi gibi çalışır.

STTouches: İki coğrafi nesnenin birbirine temas edip etmediğini test eder. Geri dönen sonuç 1 ise temas vardır.

STUnion: Coğrafi nesnelerin birleşimini geri döndürür.

STWithin: Bir coğrafi nesnenin tamamıyla diğer coğrafi nesnenin içinde olup olmadığını test eder. Geri dönen sonuç 1 ise nesne diğer nesnenin içindedir.

STX: Noktanın X koordinatını geri döndürür.

STY: Noktanın Y koordinatını geri döndürür.

AsGML: Coğrafi nesneyi GML olarak geri döndürür.

AsTextZM: Coğrafi nesneyi OGC tarafından oluşturulan WKT formatında 4 boyutlu olarak geri döndürür.

BufferWithTolerance: Bir coğrafi nesneden belirtilen uzaklıkta ki tüm noktaları kapsayacak şekilde bir kapalı alan geri döndürür.

6. SQL SERVER 2008 İÇİN SİSTEM TASARIMI

SQL Server 2008'in üzerinde çalışacak olan Konumsal Yetkilendirme sistemi; veritabanı istemcilerin erişimlerini coğrafi bölgelere göre yetkilendirmeyi amaçlamaktadır. Bu amaç doğrultusunda “Ayrık Tablolar Yöntemi” ve “Yetki Alanı Yöntemi” şeklinde adlandırılan iki farklı yöntem geliştirilmiştir. Bu yöntemlerin tasarımları yapılmış ve iki farklı sistemin birbirlerine göre üstünlükleri ve eksiklikleri incelenmiştir. Bu incelemeler sonucunda “Yetki Alanı” yöntemi birçok yönden “Ayrık Tablolar” yöntemine üstünlük sağladığından, geliştirilmek üzere bu yöntem seçildi. Bu konuyla ilgili ayrıntılar bölüm 6.4’de açıklanacaktır.

Her veritabanı sorgusunda tüm kayıtlar için “Yetki bölgesi içinde mi?” sorgusu yapılmasının performans açısından çok kötü bir sonuç oluşturacağı kesindir. Coğrafi veri üstünde yapılacak bir analiz karmaşık geometrik problemlerin çözülmesini gerektirir. Bu analiz verinin büyüklüğüne göre çok fazla zaman alabilir.

Konumsal yetkilendirme sistemi her veritabanı kaydı için “Yetkilendirme bölgesinin içinde mi?” sorgusunu yapmak zorundadır. Ayrık Tablolar ve Yetki Alanı yöntemleri bu işlemin her veritabanı sorgusunda yapılmasının önüne geçmek için geliştirildi.

Her iki yöntem de söz konusu sorguyu sadece veriler eklenirken ve konumsal verinin depolandığı alanın değişmesi durumunda hesaplayarak performansı artırmayı hedeflemektedir.

Sistem, her iki yetkilendirme yönteminin de “Çekirdek Modül” olarak adlandırılan ortak bir modül üzerinde çalışabilecek şekilde tasarlanmıştır. Bu modül konumsal yetkilendirme işlemini gerçekleştirebilmek için asgari çalışma ortamını oluşturmaktadır. Bu yüzden veritabanı yöneticisinin tablolar için tanımladığı konumsal yetki bölgelerini depolayan tabloları içerir. Ayrıca bu bölgeler ile kullanıcılar arasındaki ilişki ve ilişki şekillerini depolayan tabloları ve bunların yönetimini sağlayan veritabanı fonksiyonlarını içerir.

6.1 Çekirdek Modül

Çekirdek Modül veritabanı kullanıcılarının erişimini yetkilendirmek için gerekli bilgileri depolamak ve bu bilgileri yönetmekle sorumludur. Bu bilgiler, yetki bölgeleri ve veritabanı kullanıcıları ile yetki bölgeleri arasındaki ilişkilerdir.

Çekirdek modül her veritabanı tablosu için bu bilgileri ayrı ayrı tutar. Bunun için Çizelge 6.1’de listelenen tabloları oluşturur ve yönetir. Detayları Çizelge 6.2’de verilen KYCM

tablosu her veritabanı için sadece bir tane oluşturur. Detayları Şekil 6.3’de ve Şekil 6.4’de verilen [KYB_{{Tablo Adı}}] ve [KYBKI_{{Tablo Adı}}] tablolarını her tablo için oluşturur ve yönetir.

Çizelge 6.1 Konumsal yetkilendirme bilgilerini depolayan veritabanı tabloları

Tablo Adı	Açıklaması
[KYCM]	Konumsal yetkilendirmenin hangi tablolar için aktif hale getirildiğini depolayan veritabanı tablosu
[KYB_{{Tablo Adı}}]	Konumsal yetki bölgelerini depolayan veritabanı tablosu
[KYBKI_{{Tablo Adı}}]	Konumsal yetki bölgeleri ile veritabanı kullanıcıları arasındaki ilişkileri ve türlerini depolayan tablosu

Çizelge 6.2 KYCM tablosuna ait alanlar

Alan Adı	Veri Türü	Varsayılan Değeri	Açıklaması
Id	uniqueidentifier	(newid())	Konumsal yetkilendirme tabloları için oluşturulan kaydın eşsiz tanımlayıcısı
TabloAdı	nvarchar(64)		Tablo Adı
AlanAdı	nvarchar(64)		Konumsal yetkilendirmenin hangi tablonun hangi kaydı üstünden yapılacağını belirtir.
YetkiTuru	tinyint	0	Yetkilendirmenin türünü belirtir. 0-Sadece içinde olanları, 1- Sadece dışındakileri

Sistem [KYB_{{Tablo Adı}}] tablosundaki kayıtlar için veritabanına yapılan ekleme, düzenleme, silme ve erişim işlemlerini yönetir.

Çizelge 6.3 KYB_{{Tablo Adı}} tablosuna ait alanlar

Alan Adı	Veri Türü	Varsayılan Değeri	Açıklaması
Id	uniqueidentifier	(newid())	Yetki bölgesini eşsiz tanımlayıcı alanı
Adı	nvarchar(64)		Yetki bölgesinin adı

Turu	tinyint	0	Yetki bölgesinin türünü belirtir. 0-Geometry 1-Geography
KYBGeometry	geometry		Geometry türündeki yetki bölgesi
KYBGeography	geography		Geography türündeki yetki bölgesi

Sistem Çizelge 6.4’de ayrıntıları verilen [KYBKI_ {Tablo Adı}] tablosundaki kayıtlar için veritabanına yapılan ekleme, düzenleme, silme ve erişim işlemleri için izinleri yönetir.

Çizelge 6.4 KYBKI_ {Tablo Adı} tablosuna ait alanlar

Alan Adı	Veri Türü	Varsayılan Değeri	Açıklaması
Id	uniqueidentifier	(newid())	Yetki ilişkisi eşsiz tanımlayıcı alanı
KYBId	uniqueidentifier		Yetki bölgesini eşsiz tanımlayıcı alanı
Kullanıcı Adı	Nvarchar(64)		Veritabanı kullanıcısı Adı
KYEkle	bit		Ekleme yetkisi 0-Yok, 1-Var
KYDüzenle	bit		Düzenleme yetkisi 0-Yok, 1-Var
KYSil	bit		Silme yetkisi 0-Yok, 1-Var

Veritabanı yöneticisi Çizelge 6.5’de listelenen veritabanı prosedürlerini kullanarak yetki bölgelerini yönetir.

Çizelge 6.5 Konumsal yetkilendirme sistemi çekirdek modülü veritabanı prosedürleri

Fonksiyon Adı	Açıklaması
bolgeid YB_Ekle(tabloadi, bolge)	Bir veritabanı tablosuna yetki bölgesi eklemek için kullanılır.
sonuc YB_Düzenle(tabloadi, bolgeid,bolge)	Bir veritabanı tablosunda tanımlamış olan yetki bölgesini silmek için kullanılır.
sonuc YB_Sil(tabloadi, bolgeid)	Bir veritabanı tablosunda tanımlamış olan yetki bölgesini düzenlemek için kullanılır.

iliski YB_Kullanici_Ekle (tabloadi, bolgeid, kullaniciadi, ekle, duzenle, sil)	Yetki bölgesi ile kullanıcı ilişkisini kurmak için kullanılır.
sonuc YB_Kullanici_Duzenle (tabloadi, bolgeid, kullaniciadi, ekle, duzenle, sil)	Yetki bölgesi ile kullanıcı ilişkisini düzenlemek için kullanılır.
sonuc YB_Kullanici_Sil (tabloadi, bolgeid, kullaniciadi, ekle, duzenle, sil)	Yetki bölgesi ile kullanıcı ilişkisini silmek için kullanılır.

6.2 Ayırık Tablolar Yöntemi

Ayırık tablolar yönteminde her yetki bölgesi için adı yetki bölgesinin eşsiz tanımlayıcı ile aynı olan ve söz konusu tablo ile aynı alanlara sahip bir tablo oluşturulur. Veritabanına yapılan her ekleme işleminde kullanıcının yetkisine göre her yetki bölgesi için kayıtlar “Yetki bölgesinin içinde mi?” sorgusu yapılır. Kayıt ana tabloya eklendikten sonra sorgu sonucuna göre yetki bölgesi tablolarına da eklenir. Aynı şekilde tüm silme ve düzenleme işlemleri de ana tabloda yapıldığı gibi yetki tablolarında da yapılır.

Bu yöntemin geliştirilme sebebi, sorgulamaların ayrı tablolar üstünde yapılması sebebiyle sorgu süresinin standart bir veritabanı tablosunda yapılan sorgu süresi ile eşit olmasıdır. Bu durum erişim yapan kullanıcının bir yetki bölgesi üstünde veya birbirleri ile örtüşmeyen yetki bölgeleri üstünde haklarının olması durumunda geçerli olacaktır.

Bu yöntemde yetki bölgesi sayısı kadar tablo oluşacağı için veritabanının boyutu büyüyecektir. Bu da beraberinde SQL Server’ın hafıza gereksinimlerini artıracaktır. Veritabanının depolandığı bilgisayarın donanım özelliklerinin yeterli olması durumunda bile her ekleme, düzenleme ve silme işlemi birden fazla tablo üstünde yapılacağı için performans kaybı yaşanacaktır.

Bunun dışında birden fazla ve birbirleri ile kısmen veya tamamen örtüşen yetki bölgeleri üstünde hakları olan bir kullanıcının yapacağı sorgu alt tabloların birleştirilmesi ile oluşan tablo üstünde yapılacağı için kayıtların tekrarları oluşacaktır. Bu durum elbette giderilebilir; ancak bu işlem ciddi sistem kaynağı tüketeceğinden dolayı küçümsenmemesi gereken performans kayıplarına sebep olacaktır.

6.3 Yetki Alanı Yöntemi

Yetki alanı yönteminde konumsal yetkilendirmenin aktif hale getirildiği tablo içinde yetkilendirme işlemlerinde kullanılmak için türü XML olan “KYA” isimli bir alan oluşturulur. Tabloya üstünde yapılan ekleme ve düzenleme işlemlerinde önceden tanımlanmış olan her yetki bölgesi için “İçinde mi?” sorgusu yapılarak bu alan güncellenir. Tüm sorgulama, düzenleme ve silme işlemleri bu alanın değeri ile sınırlandırılır.

Tablo üstünde ekleme dışında yapılan tüm işlemler de ilk olarak tablo işlemi yapan kullanıcının haklarına göre XQUERY (XML sorgulama dili) ile filtrelendirir. İşlemler XQUERY sorgu sonucu üstüne uygulanır. Her işlem için ekstradan XQUERY sorgusu için gerekli hafıza ve işlem gücü gereksinimi duyacaktır. Bu gereksinim tablo içindeki verinin büyüklüğüne ve konumsal yetki bölgelerinin sayısı ile ilişkilidir. Bu yöntemin en büyük dezavantajı her sorgunun iki kez yapılmasından kaynaklı performans kaybıdır. Oluşturulan indeksler ile bu kayıp ihmal edilebilir değerlere çekilebilecektir.

Yöntemin “Ayrık Tablolar” yöntemine göre en büyük avantajı bütün işlemin tek bir tablo üstünde yapılmasından dolayı SQL Server 2008’e uyumlu geliştirilmiş tüm sistemlerle uyumlu çalışabilecek olmasıdır.

Bunun dışında kısmen ve tamamen örtüşen yetki bölgeleri üzerinde hakları olan kullanıcıların yapacağı sorgulamalarda tekrarlanan kayıtlar oluşmayacağından dolayı ciddi performans kazanımları sağlanacaktır.

6.4 Ayrık Tablolar ve Yetki Alanı Yöntemlerinin Karşılaştırılması

Her iki yöntem içinde her koşulda daha iyidir denilemez. Bu yüzden farklı uygulama alanlarına ve uygulama alanlarındaki süreçlere göre karşılaştırmalar yapılmıştır.

Bir sistemin yaşam çevrimindeki süreçlerin en başında bulunan tasarım ve geliştirme süreci içinde iki yöntemde sistemin tasarımcısı ve geliştiricisi ile doğrudan bir etkileşimi olmadığı için karşılaştırma aşamasında tasarım ve geliştirme süreçleri ihmal edilmiştir.

Karşılaştırmalar sistemin işletilmesi, bakımı ve kurtarılması süreçlerindeki uygulamaya özgü veri türlerine göre incelenmiştir. İnceleme sonuçları maliyet, performans ve risk yönetimi başlıkları altında değerlendirilmiş ve karar alınmıştır.

Veritabanı işletimi; veritabanını çalışır durumda tutmak ve uygulamanın ihtiyaçlarına cevap verecek şekilde geliştirmektir. İşletme süreçlerinde karşılaştırma yapabilmek için sürecin

eylemleri ortaya çıkarılmıştır. Buna göre veritabanı işletme sürecinde geliştirilen sistemin eylemleri ve yöntemlerin karşılaştırması Çizelge 6.6, Çizelge 6.7 ve Çizelge 6.8’de gösterilmiştir.

Çizelge 6.6Veritabanı işletme süreci eylemleri için yöntemlerin karşılaştırılması

Eylem	Ayrık Tablolar Yöntemi	Yetki Alanı Yöntemi	Değerlendirme
Geliştirme	Yavaş	Hızlı	Ayrık tablolar yönteminde Function, Trigger ve Stored Procedure geliştirmeleri her tablo için ayrı ayrı yapılmalıdır. Geliştirme eylemleri için Yetki Alanı Yöntemi daha avantajlıdır.
Veri Bakımı	Yavaş	Hızlı	Veri bakımı tekrarlı ve eskimiş verinin silinmesi veya arşivlenmesidir. Yetki alanı yöntemi ile işletilen bir veritabanında işlemler daha az zaman alacağı için Yetki Alanı Yöntemi daha avantajlıdır.
İndeks düzenleme	Yavaş	Hızlı	İndesklerin düzenlenmesi işlemi veri ekleme, güncelleme ve silme işlemleri sırasında devamlı değişen indeks kayıtlarının daha verimli hale getirilmesidir. İndekslerin her tablo için ayrı ayrı düzenlenmesi gerektiği için Yetki alanı yöntemi daha avantajlıdır.

Çizelge 6.7 Veritabanı bakım süreci eylemleri için yöntemlerin karşılaştırılması

Eylem	Ayrık Tablolar Yöntemi	Yetki Alanı Yöntemi	Değerlendirme
Yedek alma	Yavaş ve Büyük Yedek Dosyası	Hızlı ve Küçük Yedek Dosyası	Yedek alma depolanan verinin türüne göre veritabanı yönetiminin en kritik eylemi olabilir. Daha az tablo ve veri olduğu için yedek alma süresi ve yedek boyutundan dolayı Yetki Alanı Yöntemi daha avantajlıdır.
Yedeklerin depolanması	Yavaş ve Büyük Yedek Dosyası	Hızlı ve Küçük Yedek Dosyası	Yedeklerin depolanması eyleminde yedeğin boyutu hem yedeğin yazıldığı ortamı hemde yedeğin transferini etkilediği için küçük boyutlu yedeğinden dolayı Yetki Alanı Yöntemi daha avantajlıdır.
Yanlışlıkların düzeltilmesi	Yavaş	Hızlı	Hemen her durumu düşünerek uygulama geliştirsekte değişen koşullar ve insan hatalarından dolayı veritabanında veriler yanlış, eksik veya bozuk depolanabilir. Daha az tablo içermesinden dolayı yanlışlıkların oluşmasına daha az sebep olacağından ve daha hızlı olduğu için Yetki Alanı Yöntemi daha avantajlıdır.
Veri bütünlüğünün denetlenmesi	Yavaş ve Çok Parçalı Veri	Hızlı ve Az Parçalı Veri	Verinin doğruluğu kadar verinin ilişkilerinin doğruluğuda önemlidir. Sistemlerin geliştirilmesi sırasında hesap edilemeyen sorunlar veya mantıksal sorunlardan dolayı veri bütünlüğü zarar görebilir. Daha az tablo olduğu için Yetki Alanı Yöntemi daha avantajlıdır.

Çizelge 6.8 Veritabanı kurtarma süreci eylemleri için yöntemlerin karşılaştırılması

Eylem	Ayrık Tablolar Yöntemi	Yetki Alanı Yöntemi	Değerlendirme
Yedeğin Transferi	Yavaş	Hızlı	Yedeğin depolandığı yerden geri yükleneceği yere olan aktarım eylemidir. Yedeğin boyutu ile doğrudan alakalıdır. Yetki alanı yöntemi daha avantajlıdır.
Yedeğin Açılması	Yavaş	Hızlı	Yedek dosyaları çoğunlukla sıkıştırılır. Daha küçük veri boyutu olduğu için Yetki Alanı Yöntemi daha avantajlıdır.
Yedeğin Geri Yüklenmesi	Yavaş	Hızlı	Yedeğin geri yüklenmesi işlemi yine doğrudan yedek boyutu ile doğru orantılıdır. Yetki Alanı Yöntemi daha avantajlıdır.

Veritabanı işletme, bakım ve kurtarma süreçlerinin tüm alt eylemlerinde Yetki Alanı Yöntemi'nin daha avantajlı olduğu Çizelge 6.6, Çizelge 6.7 ve Çizelge 6.8'de görülmektedir. Buna rağmen Ayrık Tablolar Yöntemi'nin de avantajlı olduğu durumlar olabilir.

İşletme, bakım ve kurtarma süreçlerinin incelemesi ile Maliyet, performans ve risk yönetimi değerlendirmesi başlıkları altında Çizelge 6.9'da değerlendirme yapılmıştır.

Çizelge 6.8Yöntemlerin Maliyet, performans ve risk karşılaştırılması

Eylem	Ayrık Tablolar Yöntemi	Yetki Alanı Yöntemi	Değerlendirme
Maliyet	Çok Maliyetli	Az Maliyetli	Ayrık tablolar Yöntemi ile daha büyük veri depolandığı için daha fazla donanım kaynağına ihtiyaç vardır. Ayrıca işletme sürecinde daha fazla insana ihtiyaç olacağı için maliyet açısından Ayrık Tablolar Yöntemi dezavantajlıdır.
Performans	Performans doğrudan doğruya verinin türüne ve sistem tasarımına bağlıdır. Ancak yinede genel bir yaklaşımla karşılaştırma yapılabilir.		
Ekleme	Yavaş	Hızlı	Tek bir tabloya ekleme yapılacağı için Yetki Alanı Yöntemi daha avantajlıdır.
Düzenleme	Yavaş	Hızlı	Tek bir tablo düzenleneceği için Yetki Alanı Yöntemi daha avantajlıdır.
Silme	Yavaş	Hızlı	Tek bir tablodan silme yapılacağı için Yetki Alanı Yöntemi daha avantajlıdır.
Okuma	Hızlı	Yavaş	Daha az kayıt içeren bir tablodan okuma yapılacağı için Ayrık tablolar yöntemi daha hızlıdır.
Risk Yönetimi	Daha Çok Riskli	Daha Az Riskli	Çok fazla tablo olduğu için işletme sürecinde daha çok insan hatasına açık olacağından ve daha çok tablo içerdiğinden dolayı daha fazla veri bozulma riski taşıdığı için Ayrık tablolar yöntemi Dezavantajlıdır.

Bir çok durumda tercihlerimizi daha kaliteli ve daha ucuz çözümlerden yana kullanırız. Çok fazla kullanıcı ve okumanın yapılacağı ve bütçesi maliyetleri karşılayacağı bir projenin parçası olan sistem için Ayrık Tablolar Yöntemi daha avantajlı olacaktır. Ancak bu durum gerçekte daha az karşılan bir durum olduğu için bir çok durumda daha avantajlı olan Yetki Alanı Yöntemi geliştirilmek üzere seçilmiştir.

7. SİSTEM YÖNETİM MODÜLLERİNİN GELİŞTİRİLMESİ

Microsoft SQL Server 2008 veritabanı sunucusu standart olarak konumsal verileri depolanmasına olanak sağladığı için sistemi geliştirilmek için tercih edilmiştir.

7.1 Uygulama Veritabanının Oluşturulması

Geliştirme işlemleri için öncelikle test isimli bir veritabanı oluşturulmuştur. Veritabanını oluşturmak için EK-2'deki SQL komutları kullanılmıştır.

7.2 Yönetim Tablolarının Oluşturulması

Yapılan tasarıma bağlı kalınarak test veritabanı içinde KYCM tablosu oluşturulmuştur. KYCM tablosunun oluşturulması için EK-2'deki SQL komutları kullanılmıştır. Id alanı için newid() metodu kullanılarak oluşturulan her kayıt için eşsiz bir guid üretilmesi sağlanmıştır.

7.3 Test Tablolarının Oluşturulması

Test işlemlerini gerçekleştirmek için geometry veri türü için Tablo_Geometry tablosu oluşturulmuştur. Bu tablonun yapısının anlaşılması için SQL komutları EK-2'de verilmiştir.

Geography veri türü testleri için Tablo_Geography tablosu oluşturulmuştur. Bu tablonun yapısının anlaşılması için SQL komutları EK-2'de verilmiştir.

7.4 Veritabanı Metodlarının Geliştirilmesi

Bu işlemlerden sonra tasarımda öngörülen kayıtlı prosedürler (storedprocedures'ler) oluşturulmuştur. İlk olarak yetkilendirme bölgelerinin tanımlanması için gerekli tabloyu oluşturan metot (stored procedure) geliştirilmiştir. Bu metoda YB_Tablolari_Olustur ismi verildi. YB_Tablolari_Olustur metodu, oluşturulacak tabloların hangi tablo için kayıt depolayacağını gösteren, konumsal verinin depolandığı alanı gösteren ve yetki türünü gösteren üç parametre almaktadır. YB_Tablolari_Olustur metodu konumsal verileri depolayacak alanın türünü dikkate alarak KYCM tablosuna gerekli kaydı ekler ve ilgili [KYB_{%Tablo_Adi}] ve [KYBKI_{%Tablo_Adi}] tablolarının oluşturur. YB_Tablolari_Olustur metodunun SQL komutları EK-2'de verilmiştir.

Yetki bölgelerinin ve bunlarla ilişkili kullanıcıların bilgilerinin oluşturulacağı metodlar geliştirilmiştir. Bu metodlar;

- YB_Ekle
- YB_Duzenle
- YB_Sil
- YB_Kullanici_Ekle
- YB_Kullanici_Duzenle
- YB_Kullanici_Sil

YB_Ekle Metoduna ait SQL komutları EK-2’de verilmiştir;

7.5 Yetki Yönetimi Alt Sisteminin Geliştirilmesi

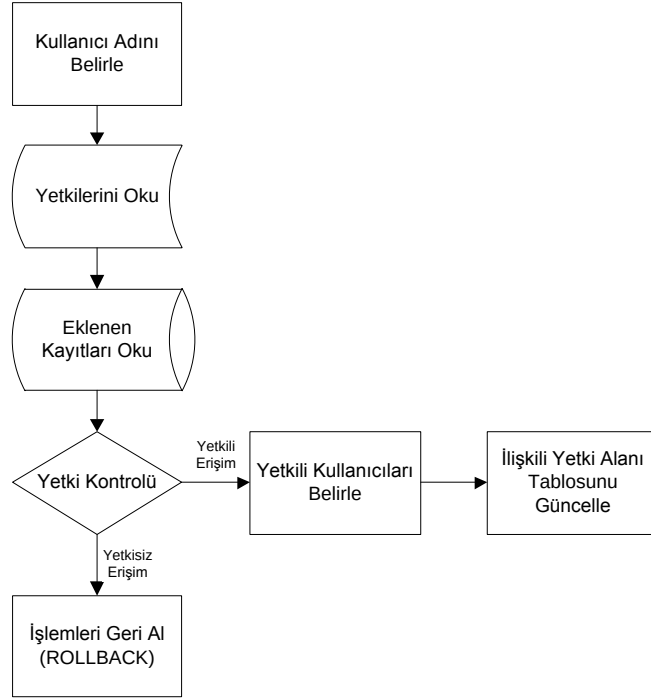
Yetki yönetimi alt sistemi birbirlerinden bağımsız çalışan iki fonksiyondan oluşur.

7.5.1 Tabloya Ekleme İşleminin Yönetilmesi

Yetki yönetimi için ilgili tabloya olan ekleme (INSERT) işlemlerini kontrol etmek için kayıt ekleme durumunda çalışan metodlar (Trigger) kullanılmıştır. Ekleme trigger’ı test tablosu üstüne kodlanmıştır. Trigger’a **trgInsert_YetkiKontrol** ismi verildi.

trgInsert_YetkiKontrol veritabanına bağlanan kullanıcının adını tespit eder. Kullanıcının yetkilerini eklenen her kayıt için kontrol eder. Eğer kayıtlardan bir tanesi bile yetki dışı ise tüm işlemi ROLLBACK komutu ile geri alır. Bu şekilde yetkisiz ekleme işlemleri engeller. Hiçbir kayıt kuralı bozmuyor ise tablo ile ilişkili yetki alanı tablolarını günceller.

Yetki alanı tablolarının güncelleme işlemi için trgInsert_YetkiKontrol Şekil 7.1’de ifade edilen işlemleri gerçekleştirir.

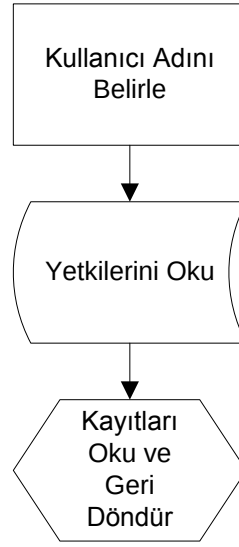


Şekil 7.1 trgInsert_YetkiKontrol akış diyagramı

7.5.2 Tablodaki Veriye Erişimin Yönetilmesi

Veriye erişim okuma, güncelleme ve silme şeklinde olacaktır. Sadece bir satırdaki veriye erişilebileceği gibi aynı anda birden çok satırdaki veriye de erişilebilir. Geliştirilecek sistem aynı anda iki durumdaki yönetebilmelidir.

Kullanıcı okuma (Select), güncelleme (Update) ve silme (Delete) işlemlerini otomatik olarak oluşan ve {%Tablo Adı}_YETKILENDIRILMIS olarak adlandırılmış olan bir table-valued function ile gerçekleştirir. Bu fonksiyon YB_Tablolari_Olustur metodunun YB_TVF_OLUSTUR metodunu çağırması ile oluşturulur. Bu fonksiyon kullanıcının yetkisi dahilinde kalan kayıtları süzerek geri döndürür. Bu şekilde güncelleme ve silme işlemleri sadece yetkilendirilmiş kayıtlar üzerinde gerçekleşmekte ve performans kaybı yaşanmamaktadır.



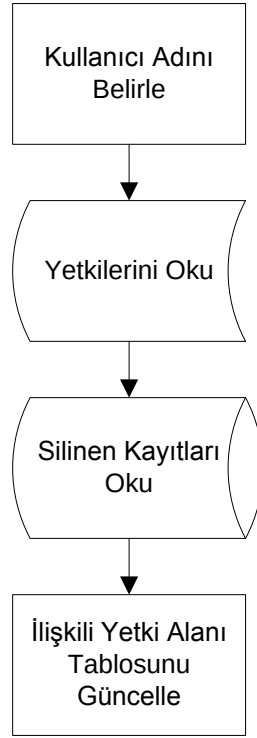
Şekil 7.2 {%Tablo Adı}_YETKILENDIRILMISakış diyagramı

Update işlemini veritabanı Delete işlemi ve ardından Insert işlemi olarak algılar. Bu sebeple güncelleme işlemi sırasında değişen her kayıt kendiliğinden `trgInsert_YetkiKontrol` tarafından yakalanarak yetki tablolarının güncellenmesi yapılacaktır. Bu sayede yetki alanı da güncellenecektir.

7.5.3 Silme İşlemi Sonrasında Kayıtların Güncellenmesi

Silme işleminin ardından ekleme işlemindeki gibi veritabanı kayıtlarının güncellenmesi yani ilişki tablolarındaki silinen kayıtlara ait kayıtların da silinmesi gerekmektedir. Bunun için `trgInsert_YetkiKontrol` metoduna benzer mantıkla çalışan bir metot geliştirilmiş ve `trgDelete_YetkiKontrol` olarak adlandırılmıştır.

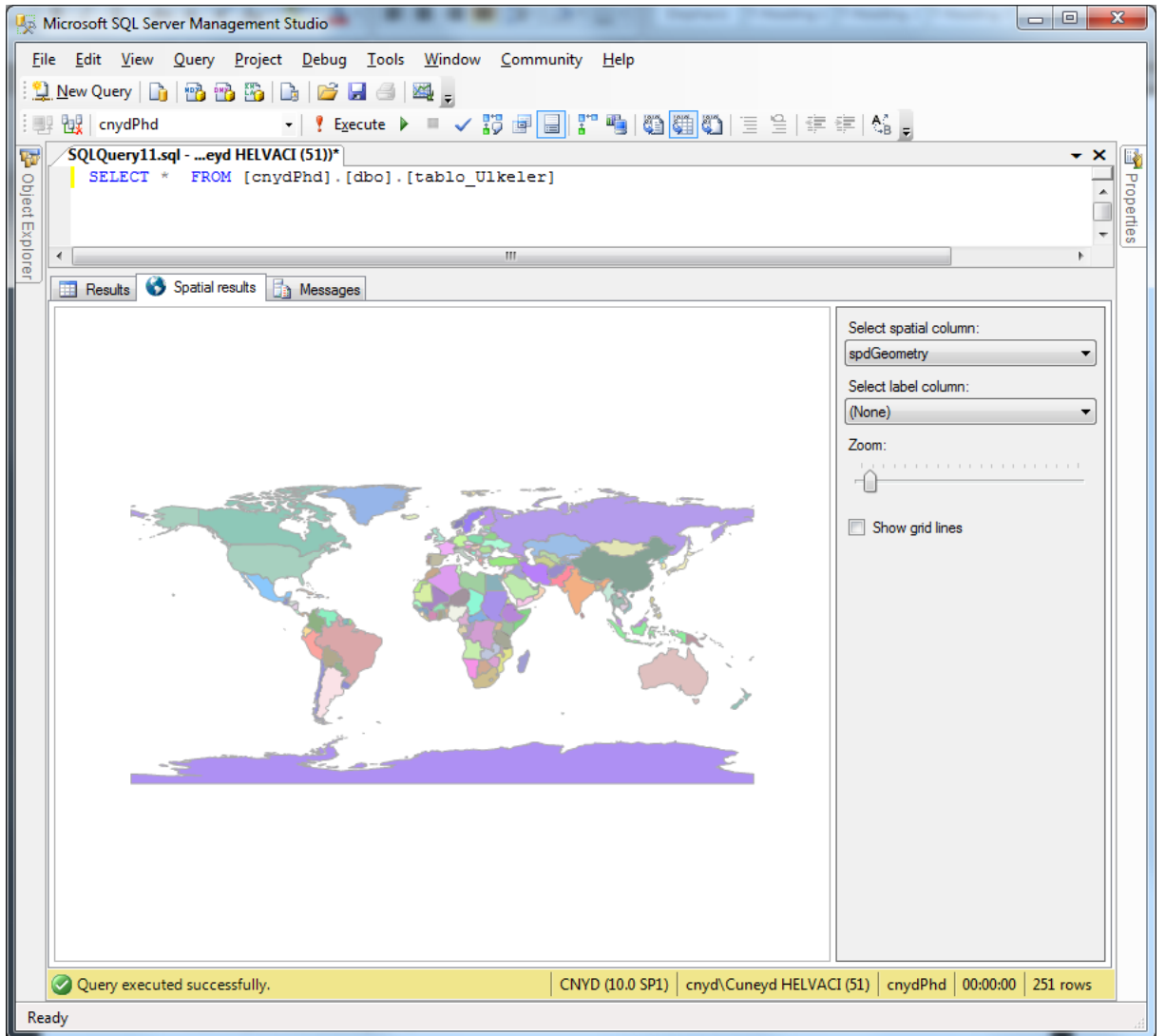
Bu metot silme işlemlerinden sonra kendiliğinden devreye girerek Şekil 7.3'teki akış diyagramındaki işlem adımlarını gerçekleştirir.



Şekil 7.3 trgInsert_YetkiKontrol akış diyagramı

8. UYGULAMANIN GERÇEK VERİLER İLE TEST EDİLMESİ

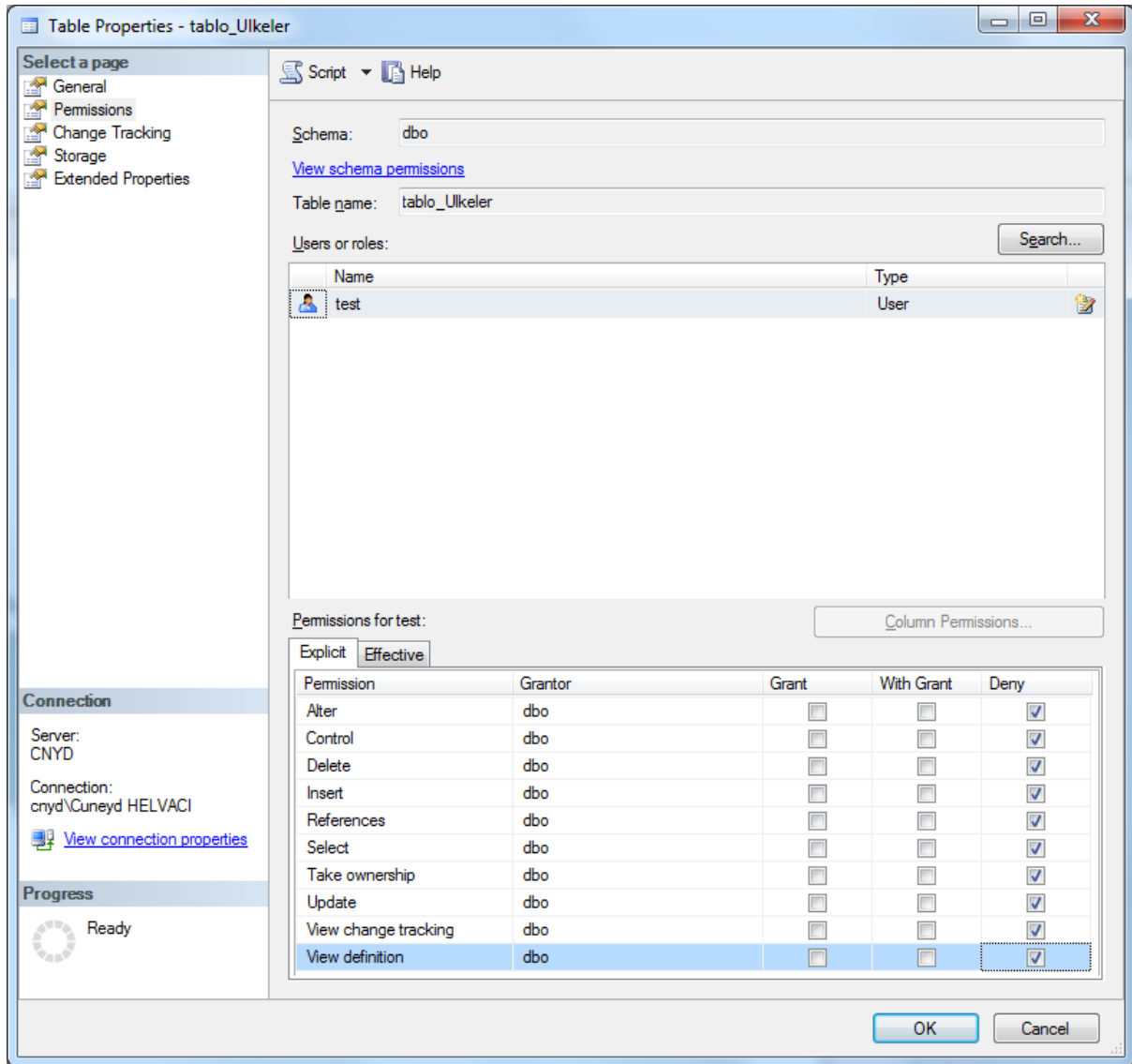
Geliştirilen uygulamanın test edilmesi için Microsoft SQL Server 2008 Management Studio isimli program kullanılmıştır. Bu program SQL Server ile kurulmaktadır. SQL Server'ı yönetmek için kullanılan başlıca programlardandır. Microsoft SQL Server 2008 Management Studio ile yapılan coğrafi bir sorgu sonucunun ekran görüntüsü Şekil 8.1'de gösterilmiştir. Bu ekran görüntüsünde tablo_Ulkeler tablosundaki tüm kayıtlar sorgulanmıştır. Sorguyu "cnyd\Cuneyd HELVACI" kullanıcısının yetkileri ile cnypdPhd veritabanı üstünde çalıştırılmıştır. Sorgu sonucu olarak 251 kayıt geri döndürmüştür.



Şekil 8.1 Microsoft SQL Server 2008 Management Studio ile yapılan ve sonucu coğrafi bir veri içeren temel bir sorgu ekran görüntüsü

Test işlemleri için her fonksiyon teker teker test edilmiştir. Yönetim fonksiyonlarını testleri için tablo_Ulkeler isimli bir tablo oluşturulmuş ve bu tablo içine “FME 2009” yazılımı ile ülke sınırlarını içeren bir Shape (shp) dosyası aktarılmıştır.

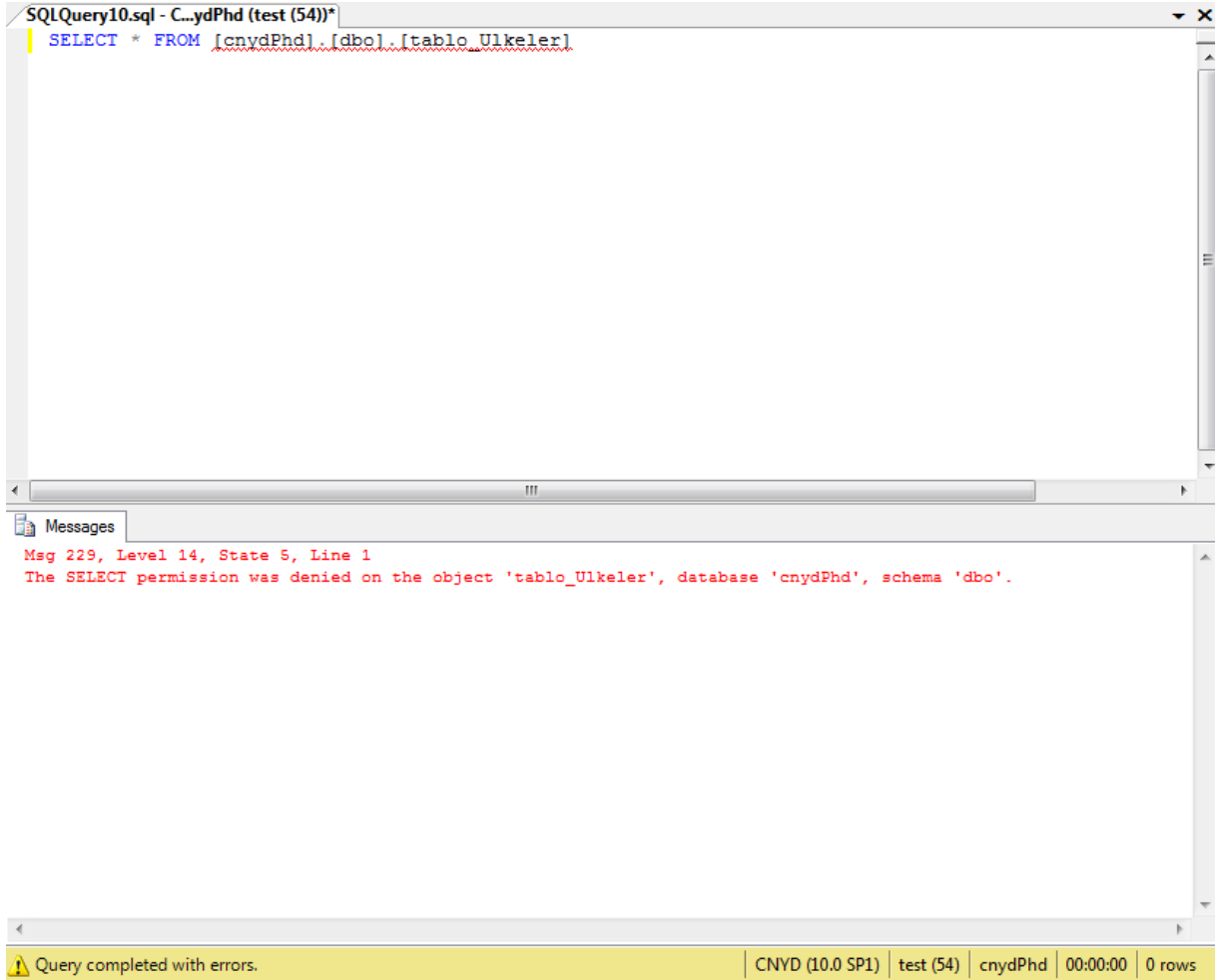
Sırası ile test isimli bir veritabanı kullanıcısı oluşturulmuştur. Bu kullanıcının tablo_Ulkeler’e olan erişimi kısıtlanmıştır. Şekil 8.2’de test kullanıcısının tablo_Ulkeler tablosuna erişimi alter, control, delete, insert, references, select, take ownership, update, view change tracking ve view definition işlemleri için “Deny” seçilerek engellenmiştir.



Şekil 8.2 Test kullanıcısının tablo_Ulkeler üstündeki yetkileri

Bunun sonucunda test kullanıcısının “Select * From tablo_Ulkeler” sorgu sonucu Şekil 8.3’de görüldüğü üzere hatayla sonuçlanmıştır. Hata açıklaması “The SELECT permission was

denied on the object 'tablo_Ulkeler', database 'cnydPhd', schema 'dbo'." şeklindedir. Bu hata mesajı test kullanıcısının cnydPhd veritabanındaki tablo_Ulkeler tablosuna erişiminin olmadığını göstermektedir.



Şekil 8.3 Test kullanıcısının tablo_Ulkeler tablosuna erişimin kısıtlandığını gösteren hata mesajı

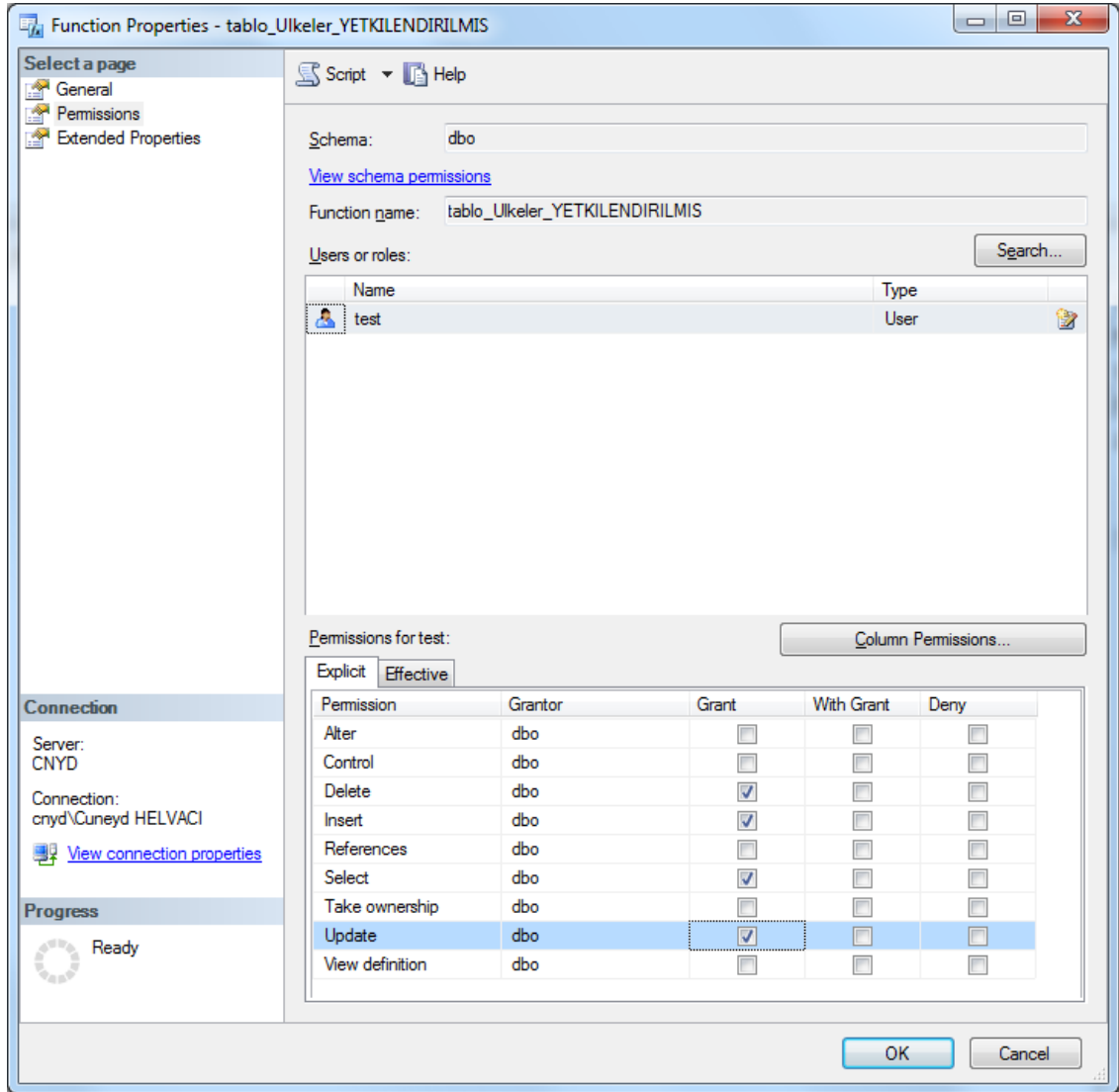
tablo_Ulkeler tablosu için konumsal yetkilendirme tablolarının oluşturulması için

```
YB_Tablolari_Olustur 'tablo_Ulkeler', 'spdGeometry', 1
```

komutu çalıştırılmış ve bulunan eksiklikler ve geliştirme hataları giderilmiştir. tablo_Ulkeler için YB_Tablolari_Olustur komutu ile konumsal erişimin denetlenmesi için tablo_Ulkeler_YETKILENDIRILMIS isimli bir table value function oluşturulmuştur.

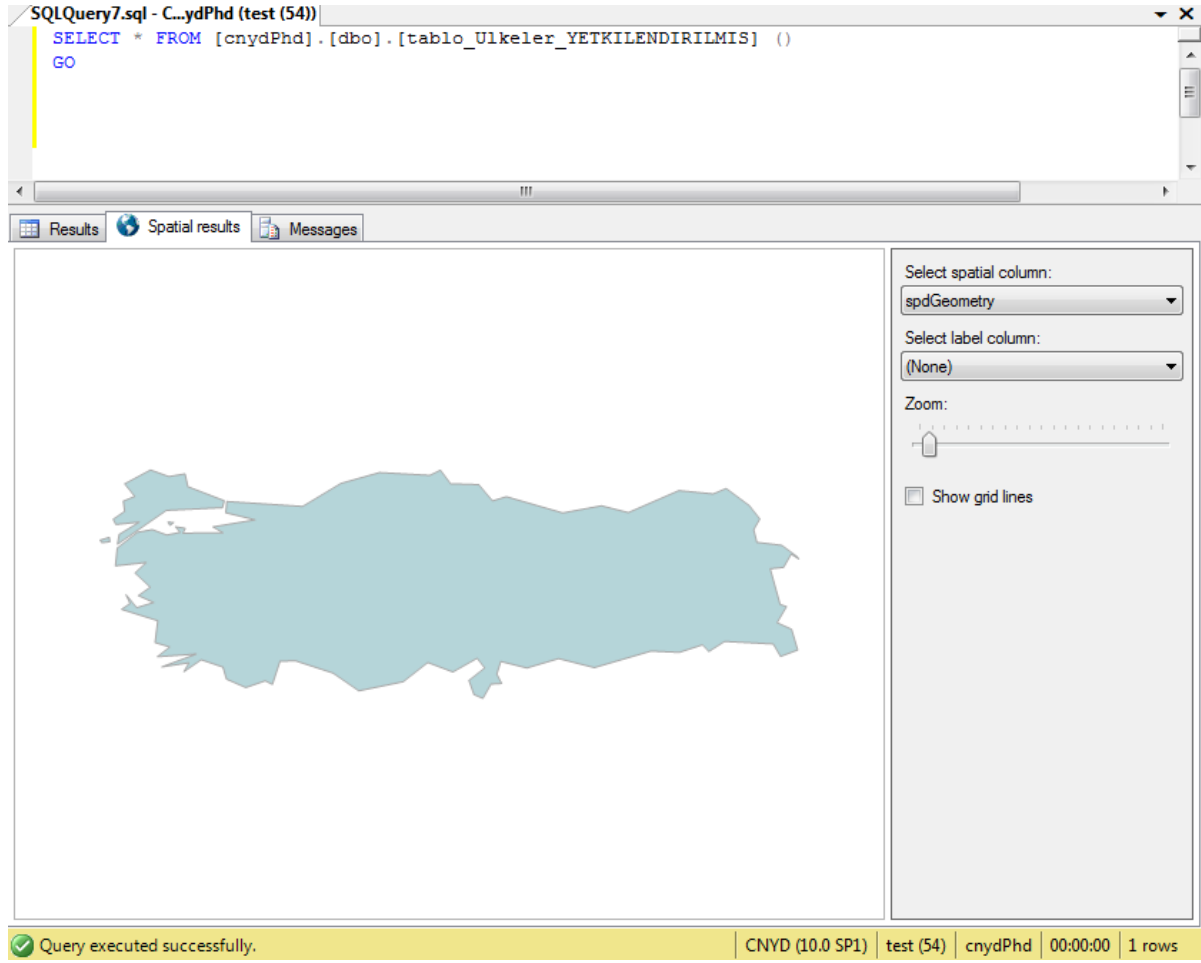
Ardından Türkiye sınırlarını içine alacak şekilde bir yetki bölgesi tanımlanmış ve test kullanıcısı için bu yetki bölgesinin içi için erişim izni verilmiştir. Ayrıca test kullanıcısına tablo_Ulkeler_YETKILENDIRILMIS fonksiyonu için Şekil 8.4'te görülen Delete, Insert,

Select ve Update yetkileri verilmiştir.



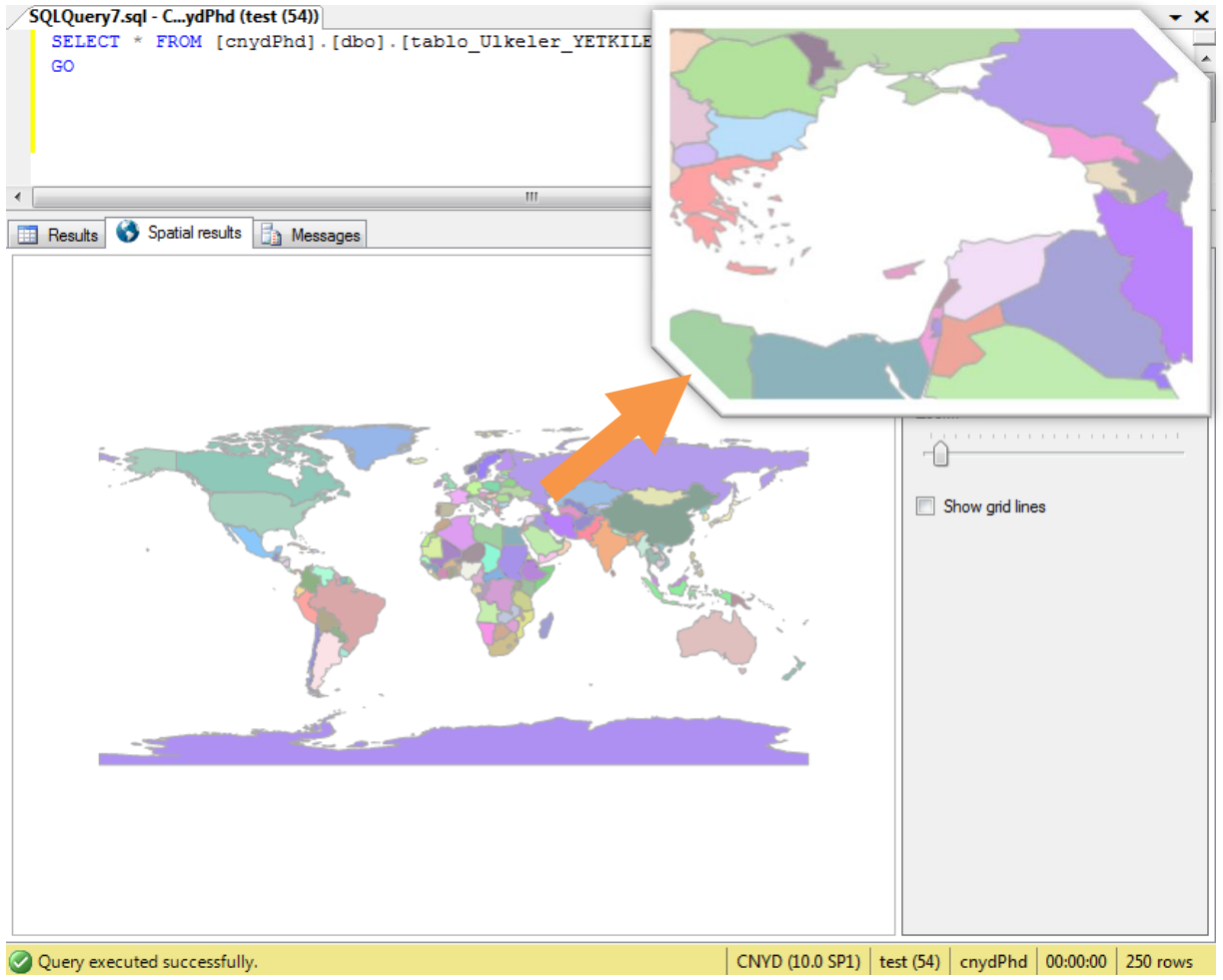
Şekil 8.4 Test kullanıcısının tablo_Ulkeler_YETKILENDIRILMIS() Fonksiyonu üstündeki yetkileri

Bunun sonucunda test kullanıcısının “Select * From tablo_Ulkeler_YETKILENDIRILMIS()” sorgusunun sonucu Şekil 8.5’de gösterilmiştir. Sorgu sonucunda, test kullanıcısının yetkileri sadece Türkiye sınırları içindeki veriye erişim hakkı verdiği için sadece Türkiye ülke sınırlarını olan bir kayda ulaşılmıştır. tablo_Ulkeler_YETKILENDIRILMIS fonksiyonu test kullanıcısının tabloda bulunan 251 kayıttan 250’sine erişimi engellemiştir.



Şekil 8.5 Test kullanıcısı tarafından çalıştırılan tablo_Ulkeler_YETKILENDIRILMIS() fonksiyonunun sonucu

Daha sonra test kullanıcısının yetki türü yetki alanının dışına erişecek şekilde değiştirildi. Bu değişikliğin ardından tablo_Ulkeler_YETKILENDIRILMIS fonksiyonu test kullanıcısının tabloda bulunan 251 kayıttan 1'ine erişimini engelleyerek 250 kayıt geri döndürmüştür. Fonksiyonun sonucu Şekil 8.6'daki gibi olmuştur ve sonuçlar içinde Türkiye yoktur.



Şekil 8.6 Test kullanıcısı tarafından çalıştırılan tablo_Ulkeler_YETKILENDIRILMIS() fonksiyonunun yeni sonucu

Bir sonraki test adımında test kullanıcısının yetkisi silindi. Bu değişikliğin ardından tablo_Ulkeler_YETKILENDIRILMIS() metodunun sonucu boş olarak geri dönmüştür.

9. PERFORMANS TESTLERİ

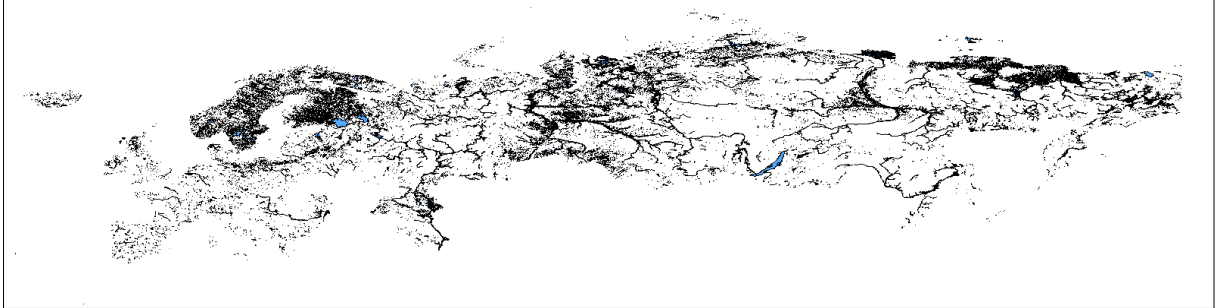
Günümüzde konumsal verileri hiçbir ek bileşene gerek kalmadan depolayan veya ek bir uygulama katmanı ile konumsal verileri barındırabilen çok sayıda veritabanı sunucusu bulunmaktadır.

Bu veritabanısunucularının hiç biri konumsal yetkilendirmeyi desteklememektedirler. Bu yüzden eşit koşullarda bir performans karşılaştırması söz konusu değildir.

Performans testleri aşağıdaki konumsal veritabanı sunucuları ve uygulamaları üzerinde yapılmıştır.

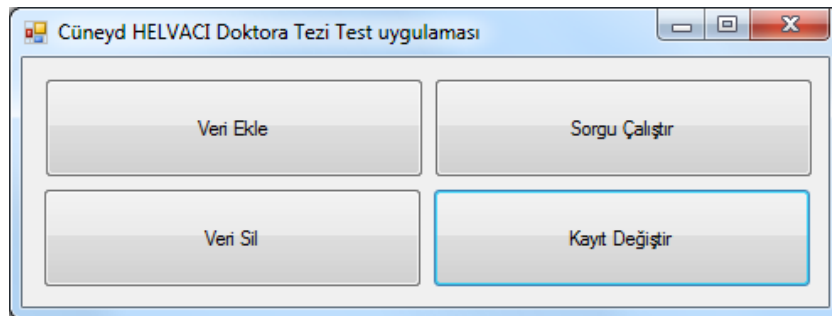
- Microsoft SQL SERVER 2008 Standart Edition
- Microsoft SQL SERVER 2008 StandartEdition(Konumsal Yetkilendirme destekli)
- Oracle 11g Standart Edition

Testler için 88188 nesneden oluşan avrupa nehirler ve ırmaklar verisi kullanılmıştır. Verinin Esri Shape dosyasının boyutu 27,8 MB'dır.Verinin kaynağı <http://resources.esri.com/arcgisonlineservices/index.cfm> web adresidir ve veri yoğunluğu Şekil 9.1'de gösterilmektedir.



Şekil 9.1 Performans testi için kullanılan veri

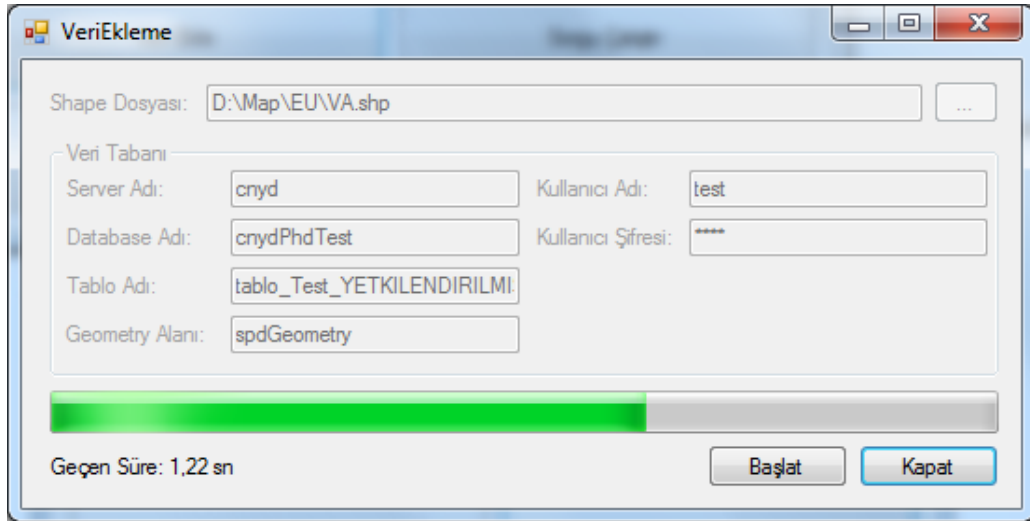
Tüm testler geliştirilen Şekil 9.2 de gösterilen özel bir arayüz ile gerçekleştirilmiştir.



Şekil 9.2 Performans test uygulaması

9.1 Ekleme Testi

Tüm ekleme operasyonları bu işlem için özel olarak yazılan bir uygulama ile gerçekleştirildi. Şekil 9.3’de ekran görüntüsü verilen uygulamaya hedef veritabanının verileri girilerek ekleme işlemleri gerçekleştirilmiştir. Çizelge 9.1’de listelenen Geliştirilen sistemin ekleme süresi olan 3,92 saniye ile Microsoft SQL SERVER 2008 Standart Edition’nın ekleme süresi olan 2,16 saniye arasındaki fark 1,76 saniyedir. Bu fark, konumsal yetkilendirme kayıtlarının oluşturulmasından kaynaklanmaktadır. Bu sonuç aynı donanım sistemi üzerinde çalışan ve 100 kat daha büyük bir veri içeren bir sistem üzerine ölçeklendiğinde aradaki fark 2 dakika 56 saniye olmaktadır. Kullanıcılar bu süreyi geliştirilen sistem ile sadece bir defa bekleyecektir. Ancak uygulama katmanında yapılan konumsal yetkilendirme bu veriyi kullanan her kullanıcının veriyi eklemek, silmek, okumak ve güncellemek için her defasında 2 dakika 56 saniye beklemek zorunda kalacaktır.



Şekil 9.3 Veri ekleme uygulaması arayüzü

Çizelge 9.1 Veri ekleme test sonuçları

Veritabanı Sunucusu	İşlem Süresi (saniye)
Microsoft SQL SERVER 2008 Standart Edition	2,16
Microsoft SQL SERVER 2008 Standart Edition (Konumsal Yetkilendirme destekli)	3,92
Oracle 11g Standart Edition	2,26

Ekleme işlemi Çizelge 9.1’de gözlemlendiği üzere güncelleme, silme ve okuma işlemlerine kıyasla daha fazla zaman almıştır. Bir veritabanı motoru ekleme işlemi daha önceden ayrılmış boş alanlara veriyi yazarak gerçekleştirir. Bu işlemin ardından tablo için tanımlanmış

indeksleri oluşturur. Eğer eklenen verinin boyutu ve indeksin boyutu veritabanındaki boş alandan fazla ise veritabanı motoru veritabanı dosyasını ihtiyacı karşılayacak kadar büyütür. Bu işlem zaman alıcıdır. Bu yüzden ekleme işleminin silme, güncelleme ve okuma işlemlerine oranla daha fazla vakit almıştır.

9.2 Güncelleme Testi

Güncelleme testi için ilk 10.000 kayıt okunmuş ve geometry alanlarındaki koordinatlara 0,000001 eklenerek güncellenmiştir.

Bir veritabanı motoru güncelleme komutunu orijinal kaydı silme ve yerine düzeltilmiş kaydı ekleme şeklinde gerçekleştirir.

Çizelge 9.2 Veri güncelleme test sonuçları

Veritabanı Sunucusu	İşlem Süresi (saniye)
Microsoft SQL SERVER 2008 Standart Edition	0,54
Microsoft SQL SERVER 2008 Standart Edition (Konumsal Yetkilendirme destekli)	0,82
Oracle 11g Standart Edition	0,50

9.3 Veri Silme Testi

Veri silme testi için 88189 kaydın tamamı “Delete From tablo” komutu ile tamamen silinmiştir. Veri silme işlemi sonrasında ekleme işleminde olduğu gibi veritabanı motoru indeksleri günceller. Konumsal yetkilendirme sistemi de benzer mantıkla yetkilendirme kayıtlarını güncellemiştir. Geliştirilen sistemin işlem süresi ile Çizelge 9.3’de listelenen diğer süreler arasındaki fark yetki kayıtlarının güncellenmesinden kaynaklanmaktadır.

Bu test örneğindeki gibi tablonun tamamının silinmesi gibi özel durumları algılayarak yetki kayıtlarını da beraberinde hiçbir koşul aramaksızın silen bir özellik gibi geliştirilen uygulamaya özel performans optimizasyonları ile çok daha iyi sonuçlar elde edilebilir.

Çizelge 9.3 Veri silme test sonuçları

Veritabanı Sunucusu	İşlem Süresi (saniye)
Microsoft SQL SERVER 2008 Standart Edition	0,06
Microsoft SQL SERVER 2008 Standart Edition (Konumsal Yetkilendirme destekli)	0,24
Oracle 11g Standart Edition	0,06

9.4 Veri Okuma Testi

Veri okuma testi için “Select * From tablo” sorgusu yapılmıştır. Okuma testi sonuçları Çizelge 9.4’de listelenmiştir. Geliştirilen sistem ile Microsoft SQL SERVER 2008 Standart Edition arasındaki fark 0,018 saniyedir. Ekleme testinde elde edilen fark 1,76 saniyedir ve bu konumsal yetkilendirme için yapılacak işlemlerin süresidir. Bu sonuçlara göre konumsal yetkilendirme işleminin geliştirilen sistem ile yapılması, uygulama katmanında yapılmasından %97,78 daha kısa sürmektedir.

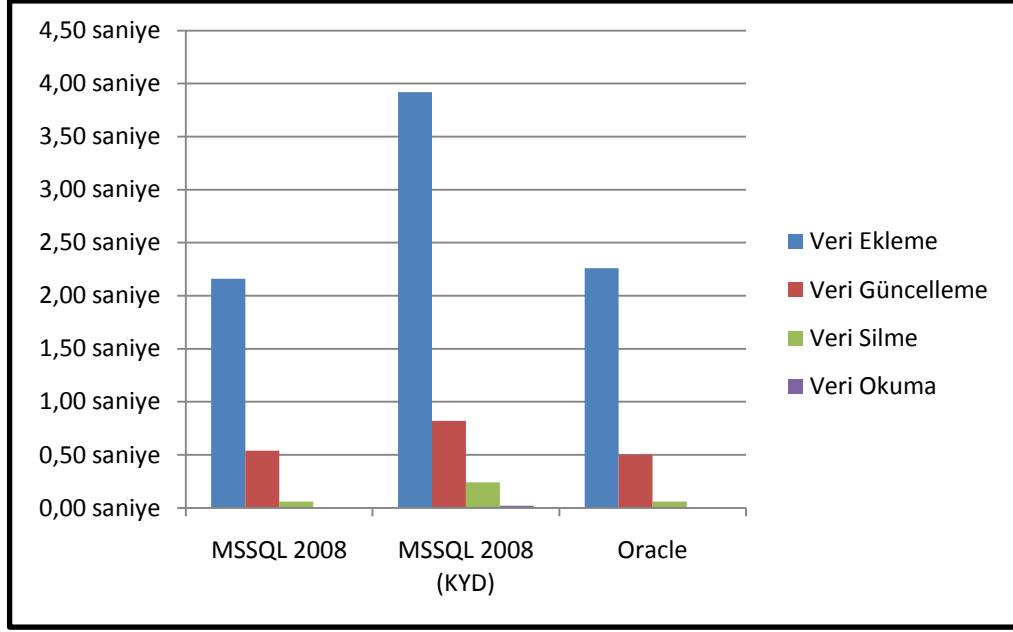
Çizelge 9.4 Veri okuma test sonuçları

Veritabanı Sunucusu	İşlem Süresi (saniye)
Microsoft SQL SERVER 2008 Standart Edition	0,004
Microsoft SQL SERVER 2008 Standart Edition (Konumsal Yetkilendirme destekli)	0,022
Oracle 11g Standart Edition	0,004

9.5 Performans Testleri Değerlendirmesi

Yapılan testler sonucunda birbirleri ile tutarlı sonuçlar elde edilmiştir. Şekil 9.4’de de görüldüğü üzere en uzun süreli sonuçlar geliştirilen konumsal yetkilendirme destekli veritabanı sunucusu ile elde edilmiştir. Bu sonuç yapılan ek işlemlerden kaynaklanmaktadır ve doğaldır.

Konumsal yetkilendirmenin gerekli olduğu bir uygulamada bu özelliğin uygulama katmanında programlanması her defasında tüm sonuçların uygulama katmanına taşınmasını gerekli kılar. Bu da gerçekleştirilen işlemin, verinin türüne ve boyutuna göre misli sonuçların gözlemlenmesi anlamına gelir.



Şekil 9.4 Genel durum grafiği

Örneğin çalışma geliştirilen uygulamada 0,018 sn gibi süren okuma işlemindeki konumsal yetki filtresi işlemini uygulama katmanında programlansaydı okunan 88189 kaydın hepsi için her okumada filtreleme işlemleri tekrarlanmak zorunda kalınacaktır. Bunun sonucunda işlemler 1,76 saniye sürecektir. Ancak testlerin yapıldığı sistemdeki verinin sadece 100 katı daha fazla veri içeren bir sistem işlem süresindeki fark 2 dakika 56 saniye olarak tahmin edilmektedir.

Konumsal yetkilendirme sisteminin performansının bağlı olduğu parametrelerin en önemlisi yetki bölgesinin nokta sayısıdır. Bu yüzden yetki bölgeleri için gerçek sınırlar kullanmak yerine genelleştirilmiş sınırlar kullanılmalıdır.

Ayrıca veritabanı için tanımlanmış başlangıç boyutu ve büyüme parametreleri ile ekleme işleminin performansı için dikkat edilmesi gereken önemli parametrelerdendir.

10. SONUÇLAR

Çalışmanın amacı ve hedefi, veritabanlarında depolanan verinin güvenliğinin sağlanması ve dolayısıyla kalitesinin artırılmasıdır. Buna paralel olarak konumsal yetkilendirme destekli uygulama geliştirmeyi mümkün kılmak için araştırmalar, tasarımlar ve geliştirmeler yapılmıştır.

Tasarım sürecinde yetkilendirme sistemini gerçekleştirebilmek için birden fazla yöntem geliştirmiştir. Bölüm 6.4’de yöntemlerin avantajları ve dezavantajları uygulama türlerine göre irdelenmiş ve Yetki Alanı Yöntemi seçilmiştir. Yapılan seçimlerin ardından uygulamalar geliştirilmiş ve test edilmiştir.

Geliştirilen uygulamanın konumsal verileri depolayabilen diğer veritabanı sunucuları ile performansları karşılaştırılmıştır. Karşılaştırma işlem süreleri ile yapılmıştır. Yapılan testlerde kullanılan diğer veritabanı sunucularının hiçbirisi konumsal yetkilendirme ile ilgili bir işlem yapmadığı için test sonuçlarının hepsinde geliştirilen sistemin daha yavaş kaldığı gözlemlenmiştir.

Ancak işlem sürelerindeki fark elbette ki uygulama katmanında yapılacak olan bir konumsal yetkilendirme işleminden çok daha kısa sürecektir. Bunun başlıca iki sebebi vardır. Bu sebeplerden birincisi süzölmüş yani daha küçük boyuttaki verinin ağ üzerinden transferinin daha kısa zaman almasıdır. Diğer sebep ise yetkilendirme işlemi ile ilgili kararın verinin veritabanına eklenmesi veya coğrafi konumunun değiştiği durumlarında yapılmasıdır. Uygulama katmanında yapılan bir konumsal yetkilendirme işleminin, istemcinin tüm veriyi kendi tarafına transfer ettikten sonra coğrafi alan ile sınırlandırarak gerçekleştirmesi ile mümkün olduğu için geliştirilen sistemle karşılaştırılması da mümkün değildir.

Performans testleri göstermiştir ki geliştirilen sistem her durumda kabul edilebilir süreler içinde işini yapmıştır. Buna rağmen sistemin eklemekten daha çok okumanın yapıldığı uygulamalarda daha performanslı çalışacaktır.

Mevcut uygulamaların bu sistemi kullanabilmeleri için yapmaları gereken tek değişiklik Microsoft SQL Server’ın bir özelliği olan Table Valued Function’ları desteklemesidir.

Ayrıca sistem kodlarının tamamının T-SQL ve .Net Framework – C# programlama dilleri ile yazılmış olmasından dolayı, sistem farklı ihtiyaçlara göre özelleştirilebilmektedir. Bu özellik ile yedekleme yükünde de ciddi tasarruflar elde edilebilecektir. Sadece çalışılan veritabanının yedeğinin alınması ile sistemin tamamının da yedeği alınmış olacaktır.

Elbette her durumda geliştirilen sistem tasarımının daha iyi olduğu söylenemez. Bunun sebebi uygulamanın gidermeye çalıştığı ihtiyaçlardaki farklılıklardır.

Bu çalışmada geliştirilen sistem büyük ölçekli verilerin yönetilmesinde faydalı olacaktır. Özellikle verinin paylaşımının denetlenmesinin yarattığı sıkıntılara çözüm getirmiştir.

Sonuç olarak veritabanı kurma, geliştirme ve işletme işlemlerinin her uygulamada farklı ihtiyaçlardan dolayı değişiklik gösterdiği gözlemlenmiştir. İyi bir veritabanı için depolanan verinin karakteristiğinin ve kullanıcıların bu veriyi nasıl sorgulayacağını çok önemli olduğu görülmüştür. Kullanıcıların sorgulama karakteri veritabanındaki verilerin indekslerin seçimi için en önemli kısıttır. İndekslerin seçimindeki kararlar veri ekleme, değiştirme, silme ve okuma sürelerini doğrudan etkileyecektir.

Veritabanlarının yedi gün yirmi dört saat çalışan bir sistem olduğu unutulmamalıdır. Bu durum beraberinde bir işletme yükü getirir. İşletme yükü bazı durumlarda doğru veritabanı tasarımları ile çok düşürülebilir. Veritabanının bakımı, düzenlemesi, ihtiyaçlara uyarlanması, yedeklenmesi gibi işlemler sistemin sürdürülebilirliğini artıracaktır. Bu işlemlerin insanlar tarafından yapıldığı göz önünde bulundurulmalı ve insan hatalarından korunmak için mümkün olduğunca standart uygulamalar tercih edilmelidir.

Veritabanı tasarımcılarının ve yöneticilerinin en büyük görevi, veritabanının ihtiyaçlara cevap vermesini sağlamaktır. Bu sebeple veritabanı tasarımcılarının ve yöneticilerinin veritabanı teknolojilerinde yaşanan gelişmeleri yakından takip etmeleri gerekmektedir.

KAYNAKLAR

- Aitchison A., (2009), "Beginning Spatial with SQL Server 2008", ISBN: 978-1-4302-1829-6.
- Frank, A. U., (1994), "Qualitative temporal reasoning in GIS – ordered time scales. In Proceedings of the 6th International Symposium on Spatial Data Handling", 410-430, Edinburg, Scotland.
- Gözüdeli Y., (2006), "Yazılımcılar için SQL Server 2005 ve Veritabanı Programlaması", ISBN: 975-02-0316 X.
- Guttman A., (1984), "R-TREES A Dynamic Index Structure for spatial Searching", University of California, Berkeley.
- Güting R.H., (1989), "An Extensible Relational Database System for Geometric Applications", Fachbereich Informatik, Universität Dortmund D-4600 Dortmund 50, West Germany
- Güting R.H., Schneider M., Hagen, (1995), "Realm-Based Spatial Data Types: The ROSE Algebra", Praktische Informatik IV, FernUniversität Hagen, Postfach 940, D-5800 Hagen, Germany.
- Lu C. T., Shekhar S. (2000), "Comparing Z-Order B-tree and R-tree Family for Spatial Query Processing", University of Minnesota, Minneapolis
- McAdams M.A. (2001), "Database Reduction Techniques For Large GIS Databases", Geosciences Department, Texas A&M University, Kingsville.
- McCurley, K.S. (2001), "Geospatial mapping and navigation of the Web", Proceedings of the 10th International World Wide Web, Hong Kong, China.
- OpenGIS, (1996), The OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing, Revision 1, OpenGIS Consortium, Inc, OpenGIS Project Document Number 96-015R1.
- OpenGIS, (1996), OpenGIS Project Document 96-025: Geodetic Reference Systems, OpenGIS Consortium, Inc, October 14.
- OpenGIS, (1999), OpenGIS Project Document 99-049: OpenGIS Simple Features Specification For SQL Revision 1.1, Open GIS Consortium, Inc. May 5, 1999.
- Petkovic D., (2006), "Microsoft SQL Server 2005", ISBN: 975-297-800-2.
- PostgreSQL Global Development Group (2002), "PostgreSQL Programmer's Guide", PostgreSQL Global Development Group, Berkeley, California.
- Ramakrishnan R., Gehrke J.E (2002), "Database Management Systems", McGraw Hill.
- Vassilakopoulos M., (2005), "Approximate Computation of Distance-Based Queries", Antonio Corral, University of Almeria, Spain, Technological Educational Institute of Thessaloniki, Greece
- Mamoulis N., Theodoridis Y., Papadias D. (2005), "Spatial Joins: Algorithms, Cost Models and Optimization", University of Hong Kong, Hong Kong, University of Piraeus, Greece, Hong Kong University of Science and Technology, Hong Kong
- Zlatanova S., (2006), "3D geometries in spatial DBMS", GIST, Delft University of Technology, The Netherlands

Internet Kaynakları:

URL 1: Esri ArcSDE Web Sayfası <http://www.esri.com/software/arcgis/arcade/>

URL 2: Microsoft SQL Server 2005 Web Sayfası <http://www.microsoft.com/sql/default.msp>

URL 3: Microsoft SQL Server 2008 Books Online <http://msdn.microsoft.com/en-us/library/ms130214.aspx>

URL 4: Oracle Spatial Engine <http://www.oracle.com/technology/products/spatial/index.html>

URL 5: PostgreSQL PostGIS Web sayfası <http://postgis.refrains.net/>

EKLER

- Ek 1 Program kodları
- Ek 2 Uygulama veritabanı SQL komutları

Ek 1 Program Kodları

CollectAggregate.cs

```
namespace SpatialExtLibrary
{
    public class ConvertGeometry
    {
        public static IGeometry ToGeometry(SqlBytes ewkb)
        {
            if (!ewkb.IsNull)
            {
                try
                {
                    MsSqlSpatialReader reader = new MsSqlSpatialReader();
                    return reader.Read(ewkb.Value);
                }
                catch { }
            }
            return null;
        }

        public static IGeometry ToGeometryFromWKB(byte[] wkb)
        {
            try
            {
                WKBReader reader = new WKBReader();
                IGeometry geometry = reader.Read(wkb);
                geometry.SRID = -1;
                return geometry;
            }
            catch { }
            return null;
        }

        internal static IGeometry ToGeometryFromWKB(SqlBytes wkb)
        {
            if (!wkb.IsNull)
            {
                try
                {
                    WKBReader reader = new WKBReader();
                    return reader.Read(wkb.Value);
                }
                catch { }
            }
            return null;
        }

        public static IGeometry ToGeometry(SqlChars wkt, SqlInt32 srid)
        {
            if (!wkt.IsNull)
            {
                IGeometry geometry =
                    new WKTRReader().Read(wkt.ToString().Value);
                geometry.SRID = (srid.IsNull ? -1 : srid.Value);
                return geometry;
            }
            return null;
        }
    }
}
```

```

        public static SqlBytes ToSqlBytes(IGeometry geometry)
        {
            if (geometry != null)
            {
                try
                {
                    return new SqlBytes(new MsSqlSpatialWriter().Write(geometry));
                }
                catch { }
            }
            return SqlBytes.Null;
        }

        public static SqlBytes ToSqlBytes(IGeometry geometry, SqlInt32 srid)
        {
            int sridValue;
            if (srid.IsNull)
                sridValue = -1;
            else
                sridValue = srid.Value;
            return ToSqlBytes(geometry, sridValue);
        }

        public static SqlBytes ToSqlBytes(IGeometry geometry, int srid)
        {
            if (geometry != null)
            {
                try
                {
                    geometry.SRID = srid;

                    return new SqlBytes(new MsSqlSpatialWriter().Write(geometry));
                }
                catch { }
            }
            return SqlBytes.Null;
        }
    }
}

```

EnvelopeAggregate.cs

```

[Serializable]
[SqlUserDefinedAggregate(Format.UserDefined
    , IsInvariantToNulls = true
    , IsInvariantToDuplicates = true
    , IsInvariantToOrder = true
    , IsNullIfEmpty = false
    , MaxByteSize = 8000
)]
public struct EnvelopeAggregate : IBinarySerialize
{
    [NonSerialized]
    private string cacheKey;
    [NonSerialized]
    private IGeometry aggregate;
}

```

```

public void Init()
{
    Aggregate.Initialize(out cacheKey, out this.aggregate);
}

public void Accumulate(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        if (this.aggregate != null)
        {
            IEnvelope pairEnvelope = geometry.EnvelopeInternal;
            IEnvelope aggregateEnvelope =
this.aggregate.EnvelopeInternal;
            aggregateEnvelope.ExpandToInclude(pairEnvelope);
            this.aggregate =
GeometryFactory.Default.ToGeometry(aggregateEnvelope);
            this.aggregate.SRID = geometry.SRID;
        }
        else
        {
            this.aggregate = geometry;
        }
    }
}

public void Merge(EnvelopeAggregate Group)
{
    if (Group.aggregate != null)
    {
        IEnvelope groupEnvelope = Group.aggregate.EnvelopeInternal;
        IEnvelope aggregateEnvelope = this.aggregate.EnvelopeInternal;
        aggregateEnvelope.ExpandToInclude(groupEnvelope);
        this.aggregate =
GeometryFactory.Default.ToGeometry(aggregateEnvelope);
        this.aggregate.SRID = Group.aggregate.SRID;
    }
}

public SqlBytes Terminate()
{
    return ConvertGeometry.ToSqlBytes(Aggregate.Terminate(cacheKey));
}

#region IBinarySerialize Members

public void Read(BinaryReader r)
{
    Aggregate.Read(r, out cacheKey, out aggregate);
}

public void Write(BinaryWriter w)
{
    Aggregate.Write(w, cacheKey, aggregate);
}

#endregion
}

```

GeometryAggregate.cs

```
namespace SpatialExtLibrary
{
    internal static class Aggregate<T>
    {
        private static readonly Dictionary<string, T> cache = new Dictionary<string,
T>();

        public static void Initialize(out string key, out T aggregate)
        {
            key = Guid.NewGuid().ToString();
            if (typeof(T).IsInterface || typeof(T).IsValueType)
            {
                aggregate = default(T);
            }
            else
            {
                aggregate = Activator.CreateInstance<T>();
            }
            cache[key] = aggregate;
        }

        public static T Terminate(string cacheKey)
        {
            T aggregate = cache[cacheKey];
            cache.Remove(cacheKey);
            return aggregate;
        }

        public static void Read(BinaryReader r, out string key, out T aggregate)
        {
            key = r.ReadString();
            aggregate = cache[key];
        }

        public static void Write(BinaryWriter w, string key, T aggregate)
        {
            w.Write(key);
            cache[key] = aggregate;
        }
    }
}
```

UnionAggregate.cs

```
[Serializable]
[SqlUserDefinedAggregate(Format.UserDefined
    , IsInvariantToNulls = true
    , IsInvariantToDuplicates = true
    , IsInvariantToOrder = true
    , IsNullIfEmpty = false
    , MaxByteSize = 8000
)]
public struct UnionAggregate : IBinarySerialize
{
    [NonSerialized]
    private string cacheKey;
    [NonSerialized]
```

```

private IGeometry aggregate;

public void Init()
{
    Aggregate.Initialize(out cacheKey, out this.aggregate);
}

public void Accumulate(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        if (this.aggregate != null)
        {
            this.aggregate = geometry.Union(this.aggregate);
            this.aggregate.SRID = geometry.SRID;
        }
        else
        {
            this.aggregate = geometry;
        }
    }
}

public void Merge(UnionAggregate Group)
{
    if (Group.aggregate != null)
    {
        this.aggregate = this.aggregate.Union(Group.aggregate);
        this.aggregate.SRID = Group.aggregate.SRID;
    }
}

public SqlBytes Terminate()
{
    return ConvertGeometry.ToSqlBytes(Aggregate.Terminate(cacheKey));
}

#region IBinarySerialize Members

public void Read(BinaryReader r)
{
    Aggregate.Read(r, out cacheKey, out aggregate);
}

public void Write(BinaryWriter w)
{
    Aggregate.Write(w, cacheKey, aggregate);
}

#endregion
}

```

GeometryColumnsTable.cs

```

namespace SpatialExtLibrary
{
    internal static class GeometryColumnsTable
    {

```

```

        internalstaticbool CheckGeometryColumnExists(SqlCommand command, string
schemaName, string tableName, string columnName)
        {
            returnHelper.CheckQueryExists(command, string.Format(
@"SELECT [F_GEOMETRY_COLUMN]
FROM [ST].[GEOMETRY_COLUMNS]
WHERE [F_TABLE_SCHEMA] = @p0
AND [F_TABLE_NAME] = @p1
AND [F_GEOMETRY_COLUMN] = @p2"
            ), schemaName
            , tableName
            , columnName
            );
        }

        internalstaticvoid ExecuteDeleteMetaRecord(SqlCommand command, string
schemaName, string tableName, string columnName)
        {
            Helper.BuildCommand(command,
@"DELETE FROM [ST].[GEOMETRY_COLUMNS]
WHERE [F_TABLE_SCHEMA] = @p0
AND [F_TABLE_NAME] = @p1
AND [F_GEOMETRY_COLUMN] = @p2"
            , schemaName
            , tableName
            , columnName
            );
            command.ExecuteNonQuery();
        }

        internalstaticvoid ExecuteAddMetaRecord(SqlCommand command, string
schemaName, string tableName, string columnName, int srid, string type)
        {
            ExecuteDeleteMetaRecord(command, schemaName, tableName,
columnName);

            Helper.BuildCommand(command, string.Format(
@"INSERT INTO [ST].[GEOMETRY_COLUMNS]
([F_TABLE_CATALOG], [F_TABLE_SCHEMA], [F_TABLE_NAME],
[F_GEOMETRY_COLUMN], [COORD_DIMENSION], [SRID], [GEOMETRY_TYPE])
VALUES ('', '{0}', '{1}', '{2}', {3}, {4}, '{5}')"
            , schemaName, tableName, columnName, 2, srid, type));
            command.ExecuteNonQuery();
        }

        internalstaticvoid ExecuteCreate(SqlCommand command)
        {
            if (!Helper.CheckMetaTableExists(command, "GEOMETRY_COLUMNS"))
            {
                command.CommandText =
@"CREATE TABLE [ST].[GEOMETRY_COLUMNS](
[F_TABLE_CATALOG] [varchar](256) NOT NULL,
[F_TABLE_SCHEMA] [varchar](256) NOT NULL,
[F_TABLE_NAME] [varchar](256) NOT NULL,
[F_GEOMETRY_COLUMN] [varchar](256) NOT NULL,
[COORD_DIMENSION] [int] NOT NULL,
[SRID] [int] NOT NULL,
[GEOMETRY_TYPE] [varchar](30) NOT NULL
)";
                command.ExecuteNonQuery();
            }
        }
    }
}

```

```

    }
}

internal static void ExecuteBackup(SqlCommand command)
{
    if (Helper.CheckMetaTableExists(command, "GEOMETRY_COLUMNS"))
    {
        string backupTableName = "ST_GEOMETRY_COLUMNS_BACKUP_" +
DateTime.Now.ToString("yyyyMMdd_HHmm");
        command.CommandText = "SELECT * INTO " + backupTableName +
" FROM [ST].[GEOMETRY_COLUMNS]";
        command.ExecuteNonQuery();
    }
}

internal static void ExecuteDeleteMetaRecords(SqlCommand command, string
schemaName, string tableName)
{
    Helper.BuildCommand(command,
@"DELETE FROM [ST].[GEOMETRY_COLUMNS]
WHERE [F_TABLE_SCHEMA] = @p0
AND [F_TABLE_NAME] = @p1"
, schemaName
, tableName
);
    command.ExecuteNonQuery();
}

internal static List<string> GetGeometryColumnsByTable(SqlCommand command,
string schemaName, string tableName)
{
    List<string> columns = newList<string>();
    Helper.BuildCommand(command,
@"SELECT [F_GEOMETRY_COLUMN]
FROM [ST].[GEOMETRY_COLUMNS]
WHERE [F_TABLE_SCHEMA] = @p0
AND [F_TABLE_NAME] = @p1"
, schemaName
, tableName
);
    using (SqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
            columns.Add(reader.GetString(0));
    }
    return columns;
}

internal static int GetSRID(SqlCommand command, string schemaName, string
tableName, string columnName)
{
    List<string> columns = newList<string>();
    Helper.BuildCommand(command,
@"SELECT [SRID]
FROM [ST].[GEOMETRY_COLUMNS]
WHERE [F_TABLE_SCHEMA] = @p0
AND [F_TABLE_NAME] = @p1
AND [F_GEOMETRY_COLUMN] = @p2
"
, schemaName
, tableName
, columnName
);
    using (SqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
            columns.Add(reader.GetInt32(0));
    }
    return columns.Count > 0 ? columns[0] : -1;
}

```

```

        , columnName
    );
    using (SqlDataReader reader = command.ExecuteReader())
    {
        if (reader.Read())
            return reader.GetInt32(0);
    }
    return -1;
}
}
}

```

GeometryTable.cs

```

namespace SpatialExtLibrary
{
    public static class GeometryTable
    {
        internal static void ExecuteCreate(SqlCommand command, string schemaName,
            string tableName, string columnName, string createColumns)
        {
            Helper.BuildCommand(command, string.Format(
@"CREATE TABLE [{0}].[{1}] (
    oid bigint IDENTITY PRIMARY KEY
    ,[{2}] varbinary(max)
    {3})"
            , schemaName, tableName, columnName, createColumns));
            command.ExecuteNonQuery();
        }

        internal static void ExecuteDropTableDependantObjects(SqlCommand command,
            string schemaName, string tableName)
        {
            if (Helper.CheckMetaTableExists(command, "GEOMETRY_COLUMNS"))
            {
                foreach (string columnName
                    in GeometryColumnsTable.GetGeometryColumnsByTable(command, schemaName, tableName))
                    SPs.ExecuteDropGeometryQueryFunctions(command,
                        schemaName, tableName, columnName);

                GeometryColumnsTable.ExecuteDeleteMetaRecords(command,
                    schemaName, tableName);
            }
        }

        internal static void ExecuteDropColumnDependantObjects(SqlCommand command,
            string schemaName, string tableName, string columnName)
        {
            SPs.ExecuteDropGeometryQueryFunctions(command, schemaName,
                tableName, columnName);
            GeometryTable.ExecuteDropTrigger(command, schemaName, tableName,
                columnName);
            GeometryTable.ExecuteDropConstraintGeometryType(command,
                schemaName, tableName, columnName);
            GeometryTable.ExecuteDropConstraintSRID(command, schemaName,
                tableName, columnName);
            GeometryTable.ExecuteDropIndexes(command, schemaName, tableName,
                columnName);
        }
    }
}

```

```

        GeometryColumnsTable.ExecuteDeleteMetaRecord(command, schemaName,
        tableName, columnName);
    }

    internalstaticvoid ExecuteAddColumn(SqlCommand command, string
    schemaName, string tableName, string columnName)
    {
        Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    ADD [{2}] varbinary(max)
    ",
        , schemaName, tableName, columnName));
        command.ExecuteNonQuery();
    }

    publicstaticvoid ExecuteAddConstraintGeometryType(SqlCommand command,
    string schemaName, string tableName, string columnName, string type)
    {
        if (!string.IsNullOrEmpty(type) && type != "GEOMETRY")
        {
            int oldTimeout = command.CommandTimeout;
            command.CommandTimeout = 0;

            Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    ADD CONSTRAINT [enforce_type_{1}_{2}]
    CHECK ([ST].[GeometryType]([{2}]) = '{3}' OR [{2}] IS NULL)"
        , schemaName, tableName, columnName, type));
            command.ExecuteNonQuery();

            command.CommandTimeout = oldTimeout;
        }
    }

    publicstaticvoid ExecuteDropConstraintGeometryType(SqlCommand command,
    string schemaName, string tableName, string columnName)
    {
        string constraintName = string.Format("enforce_type_{0}_{1}",
        tableName, columnName);
        if (Helper.CheckConstraintExists(command, schemaName, tableName,
        constraintName))
        {
            Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    DROP CONSTRAINT [{2}]"
        , schemaName, tableName, constraintName));
            command.ExecuteNonQuery();
        }
    }

    publicstaticvoid ExecuteAddConstraintSRID(SqlCommand command, string
    schemaName, string tableName, string columnName, int srid)
    {
        int oldTimeout = command.CommandTimeout;
        command.CommandTimeout = 0;

        Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    ADD CONSTRAINT [enforce_srid_{1}_{2}]
    CHECK ([{2}] IS NULL OR [ST].[SRID]([{2}]) = {3})"
        , schemaName, tableName, columnName, srid));
    }

```

```

        command.ExecuteNonQuery();
        command.CommandTimeout = oldTimeout;
    }

    public static void ExecuteDropConstraintSRID(SqlCommand command, string
schemaName, string tableName, string columnName)
    {
        string constraintName = string.Format("enforce_srid_{0}_{1}",
tableName, columnName);
        if (Helper.CheckConstraintExists(command, schemaName, tableName,
constraintName))
        {
            Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
DROP CONSTRAINT [{2}]"
, schemaName, tableName, constraintName));
            command.ExecuteNonQuery();
        }
    }

    public static void ExecuteAddIndexes(SqlCommand command, string schemaName,
string tableName, string columnName)
    {
        int oldTimeout = command.CommandTimeout;
        command.CommandTimeout = 0;

        Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
ADD
    [{2}_Envelope_MinX] float
    , [{2}_Envelope_MinY] float
    , [{2}_Envelope_MaxX] float
    , [{2}_Envelope_MaxY] float
"
, schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        Helper.BuildCommand(command, string.Format(
@"UPDATE [{0}].[{1}] SET
    [{2}_Envelope_MinX] = ST.MinX([ {2} ])
    , [{2}_Envelope_MinY] = ST.MinY([ {2} ])
    , [{2}_Envelope_MaxX] = ST.MaxX([ {2} ])
    , [{2}_Envelope_MaxY] = ST.MaxY([ {2} ])
"
, schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        Helper.BuildCommand(command, string.Format(
@"CREATE INDEX [IDX_{2}_Envelope_MinX] ON [{0}].[{1}] ([{2}_Envelope_MinX])"
, schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        Helper.BuildCommand(command, string.Format(
@"CREATE INDEX [IDX_{2}_Envelope_MinY] ON [{0}].[{1}] ([{2}_Envelope_MinY])"
, schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        Helper.BuildCommand(command, string.Format(
@"CREATE INDEX [IDX_{2}_Envelope_MaxX] ON [{0}].[{1}] ([{2}_Envelope_MaxX])"
, schemaName, tableName, columnName));
        command.ExecuteNonQuery();
    }

```

```

        Helper.BuildCommand(command, string.Format(
@"CREATE INDEX [IDX_{2}_Envelope_MaxY] ON [{0}].[{1}] ([{2}_Envelope_MaxY])"
        , schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        Helper.BuildCommand(command, string.Format(
@"CREATE INDEX [IDX_{2}_Envelope] ON [{0}].[{1}] (
        [{2}_Envelope_MinX]
        ,[{2}_Envelope_MaxX]
        ,[{2}_Envelope_MinY]
        ,[{2}_Envelope_MaxY]
        )"
        , schemaName, tableName, columnName));
        command.ExecuteNonQuery();

        command.CommandTimeout = oldTimeout;
    }

    public static void ExecuteDropIndexes(SqlCommand command, string
schemaName, string tableName, string columnName)
    {
        Helper.DropIndex(command, schemaName, tableName, "IDX_" +
columnName + "_Envelope_MinX");
        Helper.DropIndex(command, schemaName, tableName, "IDX_" +
columnName + "_Envelope_MinY");
        Helper.DropIndex(command, schemaName, tableName, "IDX_" +
columnName + "_Envelope_MaxX");
        Helper.DropIndex(command, schemaName, tableName, "IDX_" +
columnName + "_Envelope_MaxY");
        Helper.DropIndex(command, schemaName, tableName, "IDX_" +
columnName + "_Envelope");

        Helper.DropColumn(command, schemaName, tableName, columnName +
"_Envelope_MinX");
        Helper.DropColumn(command, schemaName, tableName, columnName +
"_Envelope_MinY");
        Helper.DropColumn(command, schemaName, tableName, columnName +
"_Envelope_MaxX");
        Helper.DropColumn(command, schemaName, tableName, columnName +
"_Envelope_MaxY");
    }

    private static string BuildTriggerName(string tableName, string columnName)
    {
        return "ST_GeometryTableTrigger_" + tableName + "_" + columnName;
    }

    public static void ExecuteCreateTrigger(SqlCommand command, string
schemaName, string tableName, string columnName)
    {
        string triggerName = BuildTriggerName(tableName, columnName);

        if (!Helper.CheckTriggerExists(command, schemaName, triggerName))
        {
            string primaryKey =
GeometryTable.GetPrimaryKeyColumn(command, schemaName, tableName);
            Helper.BuildCommand(command, string.Format(
@"CREATE TRIGGER [{0}].[{3}]
        ON [{0}].[{1}]
        WITH EXECUTE AS CALLER

```

```

AFTER UPDATE, INSERT
AS
BEGIN
    IF UPDATE({2})
    BEGIN
        SET NOCOUNT ON

        UPDATE [{0}].[{1}] SET
            [{2}_Envelope_MinX] = ST.MinX(inserted.[{2}])
            ,[{2}_Envelope_MinY] = ST.MinY(inserted.[{2}])
            ,[{2}_Envelope_MaxX] = ST.MaxX(inserted.[{2}])
            ,[{2}_Envelope_MaxY] = ST.MaxY(inserted.[{2}])
        FROM inserted
        INNER JOIN [{0}].[{1}]
        ON inserted.[{4}] = [{0}].[{1}].[{4}]

        SET NOCOUNT OFF
    END
END"
, schemaName
, tableName
, columnName
, triggerName
, primaryKey
));
    command.ExecuteNonQuery();
}

}

public static void ExecuteDropTrigger(SqlCommand command, string
schemaName, string tableName, string columnName)
{
    string triggerName = BuildTriggerName(tableName, columnName);
    if (Helper.CheckTriggerExists(command, schemaName, triggerName))
    {
        Helper.BuildCommand(command, string.Format(
@"DROP TRIGGER [{0}].[{1}]"
, schemaName, triggerName));
        command.ExecuteNonQuery();
    }
}

internal static string GetPrimaryKeyColumn(SqlCommand command, string
schemaName, string tableName)
{
    Helper.BuildCommand(command,
@"EXECUTE sp_pkeys @p1, @p0"
, schemaName, tableName);
    string primaryKey = null;
    using (SqlDataReader reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            primaryKey =
reader.GetString(reader.GetOrdinal("COLUMN_NAME"));
        }
    }

    if (string.IsNullOrEmpty(primaryKey))
    {
        primaryKey = tableName + "_oid";
    }
}

```

```

        Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    ADD [{2}] bigint IDENTITY PRIMARY KEY"
, schemaName, tableName, primaryKey));
        command.ExecuteNonQuery();
    }

    return primaryKey;
}

}
}

```

SpatialRefSysTable.cs

```

namespace SpatialExtLibrary
{
    public static class SpatialRefSysTable
    {
        internal static bool CheckSRIDExists(SqlCommand command, int srid)
        {
            if (srid == -1)
                return true;
            return Helper.CheckQueryExists(command,
@"SELECT [SRID] FROM [ST].[SPATIAL_REF_SYS] WHERE [SRID] = @p0"
, srid
);
        }

        internal static bool IsValidSRID(int srid)
        {
            if (srid == -1)
                return true;
            using (SqlConnection connection = Helper.ContextConnection)
            {
                connection.Open();
                using (SqlCommand command = connection.CreateCommand())
                {
                    return SpatialRefSysTable.CheckSRIDExists(command,
srid);
                }
            }
        }

        internal static string GetSRText(int srid)
        {
            if (srid == -1)
                return string.Empty;
            using (SqlConnection connection = Helper.ContextConnection)
            {
                connection.Open();
                using (SqlCommand command = new SqlCommand())
                {
                    command.Connection = connection;
                    Helper.BuildCommand(command,
@"SELECT [SRTEXT] FROM [ST].[SPATIAL_REF_SYS] WHERE [SRID] = @p0"
, srid
);
                    return (string)command.ExecuteScalar();
                }
            }
        }
    }
}

```

```

    }
}

internalstaticvoid ExecuteCreate(SqlCommand command)
{
    if (!Helper.CheckMetaTableExists(command, "SPATIAL_REF_SYS"))
    {
        command.CommandText =
@"CREATE TABLE [ST].[SPATIAL_REF_SYS](
    [SRID] [int] NOT NULL,
    [AUTH_NAME] [varchar](256) NULL,
    [AUTH_SRID] [int] NULL,
    [SRTEXT] [varchar](2048) NULL,
    PRIMARY KEY CLUSTERED ([SRID] ASC)
    WITH (PAD_INDEX = OFF, IGNORE_DUP_KEY = OFF)
)";
        command.ExecuteNonQuery();

        ExecutePopulate(command);

        ExecuteCreateRefreshTransfromCacheTrigger(command);
    }
}

internalstaticvoid ExecuteBackup(SqlCommand command)
{
    if (Helper.CheckMetaTableExists(command, "SPATIAL_REF_SYS"))
    {
        string backupTableName = "ST_SPATIAL_REF_SYS_BACKUP_" +
DateTime.Now.ToString("yyyyMMdd_HHmm");
        command.CommandText = "SELECT * INTO " + backupTableName +
" FROM [ST].[SPATIAL_REF_SYS]";
        command.ExecuteNonQuery();
    }
}

publicstaticvoid ExecuteCreateRefreshTransfromCacheTrigger(SqlCommand
command)
{
    if (Helper.CheckMetaTableExists(command, "SPATIAL_REF_SYS"))
    {
        Type type = typeof(CustomFunctions);
        Helper.BuildCommand(command, string.Format(
@"CREATE TRIGGER [{0}].[{1}]
ON [{0}].[{2}]
WITH EXECUTE AS CALLER
FOR UPDATE, DELETE
AS EXTERNAL NAME [{3}].[{4}]{5}].[{1}]"
, "ST"
, "RefreshTransformCacheTrigger"
, "SPATIAL_REF_SYS"
, type.Assembly.GetName().Name
, (type.Namespace == null ? "" : type.Namespace + ".")
, type.Name
));
        command.ExecuteNonQuery();
    }
}

```

```

        public static void ExecuteDropRefreshTransfromCacheTrigger(SqlCommand
command)
    {
        string schemaName = "ST";
        string triggerName = "RefreshTransformCacheTrigger";
        if (Helper.CheckTriggerExists(command, schemaName, triggerName))
        {
            Helper.BuildCommand(command, string.Format(
@"DROP TRIGGER [{0}].[{1}]"
, schemaName
, triggerName
));
            command.ExecuteNonQuery();
        }
    }

    internal static void ExecutePopulate(SqlCommand command)
    {
        Helper.BuildCommand(command,
@"INSERT INTO [ST].[SPATIAL_REF_SYS]
VALUES (@SRID,'EPSG',@SRID,@SRTEXT)"
);
        command.Parameters.Add("@SRID", SqlDbType.Int, -1);
        command.Parameters.Add("@SRTEXT", SqlDbType.VarChar, -1);

        command.Prepare();

        Stream resource =
System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceStream("SpatialEx
tLibrary.SRID.csv");
        using (StreamReader sr = new StreamReader(resource))
        {
            while (!sr.EndOfStream)
            {
                string line = sr.ReadLine();
                int split = line.IndexOf(';');
                if (split > -1)
                {
                    string srid = line.Substring(0, split);
                    string wkt = line.Substring(split + 1);
                    command.Parameters["@SRID"].Value =

int.Parse(srid);

                    command.Parameters["@SRTEXT"].Value = wkt;
                    command.ExecuteNonQuery();
                }
            }
            command.Parameters.Clear();
        }
    }
}

```

ConvertGeometry.cs

```

namespace SpatialExtLibrary
{
    public class ConvertGeometry
    {
        public static IGeometry ToGeometry(SqlBytes ewkb)

```

```

    {
        if (!ewkb.IsNull)
        {
            try
            {
                MsSqlSpatialReader reader = new MsSqlSpatialReader();
                return reader.Read(ewkb.Value);
            }
            catch { }
        }
        return null;
    }

    public static IGeometry ToGeometryFromWKB(byte[] wkb)
    {
        try
        {
            WKBReader reader = new WKBReader();
            IGeometry geometry = reader.Read(wkb);
            geometry.SRID = -1;
            return geometry;
        }
        catch { }
        return null;
    }

    internal static IGeometry ToGeometryFromWKB(SqlBytes wkb)
    {
        if (!wkb.IsNull)
        {
            try
            {
                WKBReader reader = new WKBReader();
                return reader.Read(wkb.Value);
            }
            catch { }
        }
        return null;
    }

    public static IGeometry ToGeometry(SqlChars wkt, SqlInt32 srid)
    {
        if (!wkt.IsNull)
        {
            IGeometry geometry =
                new WKTRReader().Read(wkt.ToString().Value);
            geometry.SRID = (srid.IsNull ? -1 : srid.Value);
            return geometry;
        }
        return null;
    }

    public static SqlBytes ToSqlBytes(IGeometry geometry)
    {
        if (geometry != null)
        {
            try
            {
                return new SqlBytes(new MsSqlSpatialWriter().Write(geometry));
            }

```

```

        catch { }
    }
    returnSqlBytes.Null;
}

public static SqlBytes ToSqlBytes(IGeometry geometry, SqlInt32 srid)
{
    int sridValue;
    if (srid.IsNull)
        sridValue = -1;
    else
        sridValue = srid.Value;
    return ToSqlBytes(geometry, sridValue);
}

public static SqlBytes ToSqlBytes(IGeometry geometry, int srid)
{
    if (geometry != null)
    {
        try
        {
            geometry.SRID = srid;

            return new SqlBytes(new MsSqlSpatialWriter().Write(geometry));
        }
        catch { }
    }
    return SqlBytes.Null;
}
}
}

```

CustomFunctions.cs

```

public sealed partial class CustomFunctions
{
    #region Constructors
    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes PolygonFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
    {
        IPolygon geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as IPolygon;
        return ConvertGeometry.ToSqlBytes(geometry, srid);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes PolygonFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IPolygon;
        return ConvertGeometry.ToSqlBytes(geometry);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes Polygon(SqlBytes ewkb, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry is ILineString)
        {
            return ConvertGeometry.ToSqlBytes(new Polygon(new LinearRing(geometry.Coordinates)),
            srid);
        }
    }
}

```

```

    }
elseif (geometry is ILinearRing)
{
return ConvertGeometry.ToSqlBytes(new Polygon((LinearRing)geometry), srid);
}
return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes PointFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
{
IPoint geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as IPoint;
return ConvertGeometry.ToSqlBytes(geometry, srid);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes PointFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
{
IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IPoint;
return ConvertGeometry.ToSqlBytes(geometry);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes Point(double x, double y, [DefaultValue(-1)] SqlInt32 srid)
{
return ConvertGeometry.ToSqlBytes(new Point(x, y), srid);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes MPolyFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
{
IMultiPolygon geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as IMultiPolygon;
return ConvertGeometry.ToSqlBytes(geometry, srid);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes MPolyFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
{
IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IMultiPolygon;
return ConvertGeometry.ToSqlBytes(geometry);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes MPointFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
{
IMultiPoint geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as IMultiPoint;
return ConvertGeometry.ToSqlBytes(geometry, srid);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes MPointFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
{
IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IMultiPoint;
return ConvertGeometry.ToSqlBytes(geometry);
}

[SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
public static SqlBytes MLineFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
{
IMultiLineString geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as IMultiLineString;
return ConvertGeometry.ToSqlBytes(geometry, srid);
}

```

```

    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes MLineFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IMultiLineString;
        return ConvertGeometry.ToSqlBytes(geometry);
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlBytes MakeEnvelope(SqlDouble X1, SqlDouble Y1, SqlDouble X2, SqlDouble
    Y2, SqlInt32 srid)
    {
        IGeometry envelope = GeometryFactory.Default.ToGeometry(new Envelope(X1.Value,
        X2.Value, Y1.Value, Y2.Value));
        return ConvertGeometry.ToSqlBytes(envelope, srid.Value);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes LineFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
    {
        ILineString geometry = ConvertGeometry.ToGeometryFromWKB(wkb) as ILineString;
        return ConvertGeometry.ToSqlBytes(geometry, srid);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes LineFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as ILineString;
        return ConvertGeometry.ToSqlBytes(geometry);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes GeomFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometryFromWKB(wkb);
        return ConvertGeometry.ToSqlBytes(geometry, srid);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes GeomFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid);
        return ConvertGeometry.ToSqlBytes(geometry);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes GeomCollFromWKB(SqlBytes wkb, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometryCollection geometry = ConvertGeometry.ToGeometryFromWKB(wkb)
        as IGeometryCollection;
        return ConvertGeometry.ToSqlBytes(geometry, srid);
    }

    [SqlFunction(IsDeterministic = true, DataAccess = DataAccessKind.Read)]
    public static SqlBytes GeomCollFromText(SqlChars wkt, [DefaultValue(-1)] SqlInt32 srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(wkt, srid) as IGeometryCollection;
        return ConvertGeometry.ToSqlBytes(geometry);
    }
    #endregion

```

```

        #region Modifiers
        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes LineMerge(SqlBytes ewkb)
        {
            IGeometryCollection geometry = ConvertGeometry.ToGeometry(ewkb) as IGeometryCollection;
            if (geometry != null)
            {
                LineMerger lineMerger = new LineMerger();
                lineMerger.Add(geometry.Geometries);
                List<ILineString> mergedLineStrings = newList<ILineString>();
                foreach (ILineString lineString in lineMerger.MergedLineStrings)
                    mergedLineStrings.Add(lineString);

                if (mergedLineStrings.Count == 1)
                    return ConvertGeometry.ToSqlBytes(mergedLineStrings[0]);
                else
                    return ConvertGeometry.ToSqlBytes(new GeometryCollection(mergedLineStrings.ToArray()));
            }
            else
                return ewkb;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes Normalize(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                geometry.Normalize();
                return ConvertGeometry.ToSqlBytes(geometry);
            }
            return SqlBytes.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes Polygonize(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                Polygonizer polygonizer = new Polygonizer();
                if (geometry is IGeometryCollection)
                    polygonizer.Add((geometry as IGeometryCollection).Geometries);
                else
                    polygonizer.Add(geometry);
                List<IPolygon> polygons = newList<IPolygon>();
                foreach (IPolygon polygon in polygonizer.Polygons)
                    polygons.Add(polygon);

                return ConvertGeometry.ToSqlBytes(new GeometryCollection(polygons.ToArray()),
                    geometry.SRID);
            }
            return ConvertGeometry.ToSqlBytes(GeometryCollection.Empty);
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes Reverse(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)

```

```

        {
            if (geometry is ILineString)
                return ConvertGeometry.ToSqlBytes((geometry as ILineString).Reverse());
            if (geometry is IMultiLineString)
                return ConvertGeometry.ToSqlBytes((geometry as IMultiLineString).Reverse());
        }
        return SqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = false, DataAccess = DataAccessKind.Read)]
    public static SqlBytes SetSRID(SqlBytes ewkb, SqlInt32 new_srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            if (geometry.SRID == new_srid.Value)
                return ewkb;
            else
                return ConvertGeometry.ToSqlBytes(geometry, new_srid.Value);
        }
        return SqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlBytes Simplify(SqlBytes ewkb, SqlDouble tolerance)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            geometry = DouglasPeuckerSimplifier.Simplify(geometry, tolerance.Value);
            if (geometry.IsEmpty)
                return SqlBytes.Null;
            return ConvertGeometry.ToSqlBytes(geometry);
        }
        return SqlBytes.Null;
    }

    private static readonly Dictionary<TransformCacheKey, IMathTransform> transformCache =
        new Dictionary<TransformCacheKey, IMathTransform>(new TransformCacheKey());
    private static readonly CoordinateSystemFactory coordinateSystemFactory =
        new CoordinateSystemFactory();
    private static readonly CoordinateTransformationFactory coordinateTransformationFactory =
        new CoordinateTransformationFactory();

    [SqlFunction(IsDeterministic = false, DataAccess = DataAccessKind.Read)]
    public static SqlBytes Transform(SqlBytes ewkb, SqlInt32 target_srid)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            int sourceSRID = geometry.SRID;
            int targetSRID = target_srid.Value;

            if (sourceSRID == targetSRID)
                return ewkb;
            else
            {
                TransformCacheKey cacheKey = new TransformCacheKey(sourceSRID, targetSRID);
                IMathTransform mathTransform;
                if (!transformCache.TryGetValue(cacheKey, out mathTransform))
                {

```

```

if (!SpatialRefSysTable.IsValidSRID(targetSRID))
    throw new InvalidSRIDException(targetSRID);

string sourceSRWKT = SpatialRefSysTable.GetSRText(sourceSRID);
string targetSRWKT = SpatialRefSysTable.GetSRText(targetSRID);
ICoordinateSystem sourceCS = coordinateSystemFactory.CreateFromWkt(sourceSRWKT);
ICoordinateSystem targetCS = coordinateSystemFactory.CreateFromWkt(targetSRWKT);
ICoordinateTransformation transform =
    coordinateTransformationFactory.CreateFromCoordinateSystems(sourceCS, targetCS);
    mathTransform = transform.MathTransform;
    transformCache[cacheKey] = mathTransform;
    }
IGeometry sourceGeometry = geometry;
IGeometry targetGeometry = GeometryTransform.TransformGeometry(sourceGeometry,
    mathTransform);
return ConvertGeometry.ToSqlBytes(targetGeometry, targetSRID);
    }
    }
return SqlBytes.Null;
    }

public static void RefreshTransformCacheTrigger()
{
    SqlTriggerContext triggerContext = SqlContext.TriggerContext;
    using (SqlConnection connection = Helper.ContextConnection)
    {
        connection.Open();
        using (SqlCommand command = connection.CreateCommand())
        {
            int srid = -1;
            if (triggerContext.TriggerAction == TriggerAction.Update
                && triggerContext.IsUpdatedColumn(3))
            {
                Helper.BuildCommand(command, "SELECT srid FROM inserted");
                try
                {
                    srid = (int)command.ExecuteScalar();
                }
                catch { }
            }
            elseif (triggerContext.TriggerAction == TriggerAction.Delete)
            {
                Helper.BuildCommand(command, "SELECT srid FROM deleted");
                try
                {
                    srid = (int)command.ExecuteScalar();
                }
                catch { }
            }

            if (srid != -1)
            {
                List<TransformCacheKey> removeKeys = newList<TransformCacheKey>();
                foreach (TransformCacheKey cacheKey in transformCache.Keys)
                {
                    if (cacheKey.part1 == srid || cacheKey.part2 == srid)
                    {
                        removeKeys.Add(cacheKey);
                    }
                }
            }
            foreach (TransformCacheKey cacheKey in removeKeys)

```

```

        {
            transformCache.Remove(cacheKey);
        }
    }
}

}

private class TransformCacheKey : CompositeKey<int, int>,
    IEqualityComparer<TransformCacheKey>
{
    public TransformCacheKey() : base() { }
    public TransformCacheKey(int sourceSRID, int targetSRID) : base(sourceSRID,
        targetSRID) { }

    public bool Equals(TransformCacheKey x, TransformCacheKey y)
    {
        return base.Equals(x, y);
    }

    public int GetHashCode(TransformCacheKey obj)
    {
        return base.GetHashCode(obj);
    }
}

private class CompositeKey<TPart1, TPart2> : IEqualityComparer<CompositeKey<TPart1,
    TPart2>>
{
    public TPart1 part1;
    public TPart2 part2;

    public CompositeKey()
    {
    }

    public CompositeKey(TPart1 part1, TPart2 part2)
    {
        this.part1 = part1;
        this.part2 = part2;
    }

    public bool Equals(CompositeKey<TPart1, TPart2> x, CompositeKey<TPart1, TPart2> y)
    {
        return x.part1.Equals(y.part1) && x.part2.Equals(y.part2);
    }

    public int GetHashCode(CompositeKey<TPart1, TPart2> obj)
    {
        return obj.part1.GetHashCode() ^ obj.part2.GetHashCode();
    }
}

#endregion

#region LinearReferencing
[SqlFunction(IsDeterministic = true)]

```

```

publicstatic SqlBytes LineInterpolatePoint(SqlBytes ewkb, SqlDouble location)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry is LineString && !location.IsNull)
    {
        LengthIndexedLine indexedLine = new LengthIndexedLine(geometry);
        double index = location.Value * geometry.Length;
        return ConvertGeometry.ToSqlBytes(new Point(indexedLine.ExtractPoint(index)),
            geometry.SRID);
    }
    return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstatic SqlDouble LineLocatePoint(SqlBytes ewkbLineString, SqlBytes ewkbPoint)
{
    IGeometry pairLineString = ConvertGeometry.ToGeometry(ewkbLineString);
    if (pairLineString is ILineString)
    {
        IPoint pairPoint = ConvertGeometry.ToGeometry(ewkbPoint) as IPoint;
        if (pairPoint != null)
        {
            LengthIndexedLine indexedLine = new LengthIndexedLine(pairLineString);
            double index = indexedLine.Project(pairPoint.Coordinate);
            return new SqlDouble(index / pairLineString.Length);
        }
    }
    return SqlDouble.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstatic SqlBytes LineSubstring(SqlBytes ewkb, SqlDouble start, SqlDouble end)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null && !start.IsNull && !end.IsNull)
    {
        LengthIndexedLine indexedLine = new LengthIndexedLine(geometry);
        double length = geometry.Length;
        double startIndex = start.Value * length;
        double endIndex = end.Value * length;
        return ConvertGeometry.ToSqlBytes(indexedLine.ExtractLine(startIndex, endIndex),
            geometry.SRID);
    }
    return SqlBytes.Null;
}
#endregion

#region Other
[SqlFunction(IsDeterministic = true)]
publicstatic SqlBytes AsBinary(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        return new SqlBytes(geometry.AsBinary());
    }
    return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstatic SqlChars AsGML(SqlBytes ewkb)

```

```

    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            XmlDocument document = new XmlDocument();
            Geometry g = (Geometry)geometry;
            document.Load(g.ToGMLFeature());
            string gml = document.FirstChild.InnerXml;

            gml = gml.Replace("xmlns:gml=\"urn:gml\"", "");

            return new SqlChars(gml);
        }
        return SqlChars.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlChars AsText(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            return new SqlChars(geometry.AsText());
        }
        return SqlChars.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlInt32 Dimension(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            return new SqlInt32((int)geometry.Dimension);
        }
        return 0;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlDouble Y(SqlBytes ewkb)
    {
        IPoint geometry = ConvertGeometry.ToGeometry(ewkb) as IPoint;
        if (geometry != null)
        {
            return new SqlDouble(geometry.Y);
        }
        return SqlDouble.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlDouble X(SqlBytes ewkb)
    {
        IPoint geometry = ConvertGeometry.ToGeometry(ewkb) as IPoint;
        if (geometry != null)
        {
            return new SqlDouble(geometry.X);
        }
        return SqlDouble.Null;
    }

    [SqlFunction(IsDeterministic = true)]

```

```

public static SqlBytes StartPoint(SqlBytes ewkb)
{
    ILineString geometry = ConvertGeometry.ToGeometry(ewkb) as ILineString;
    if (geometry != null)
    {
        return ConvertGeometry.ToSqlBytes(geometry.StartPoint);
    }
    return SqlBytes.Null;
}

[SqlFunction]
public static SqlInt32 SRID(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        return new SqlInt32(geometry.SRID);
    }
    return new SqlInt32(-1);
}

[SqlFunction(IsDeterministic = true)]
public static SqlBytes PointN(SqlBytes ewkb, SqlInt32 index)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        try
        {
            if (geometry is ILineString)
            {
                return ConvertGeometry.ToSqlBytes((geometry as ILineString).GetPointN(index.Value - 1));
            }
        }
        catch { }
    }
    return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlInt32 NumPoints(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        if (geometry is IPoint || geometry is IMultiPoint)
            return 1;
        elseif (geometry is IGeometryCollection)
            return (geometry as IGeometryCollection).Geometries[0].NumPoints;
        else
            return geometry.NumPoints;
    }
    return SqlInt32.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlInt32 NumInteriorRings(SqlBytes ewkb)
{
    IPolygon geometry = ConvertGeometry.ToGeometry(ewkb) as IPolygon;
    if (geometry != null)
    {
        return new SqlInt32(geometry.NumInteriorRings);
    }
}

```

```

returnSqlInt32.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlInt32 NumGeometries(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnnewSqlInt32(geometry.NumGeometries);
        }
        returnSqlInt32.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBoolean IsSimple(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnnewSqlBoolean(geometry.IsSimple);
        }
        returnSqlBoolean.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBoolean IsRing(SqlBytes ewkb)
    {
        ILineString geometry = ConvertGeometry.ToGeometry(ewkb) asILineString;
        if (geometry != null)
        {
            returnnewSqlBoolean(geometry.IsRing);
        }
        returnSqlBoolean.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBoolean IsEmpty(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnnewSqlBoolean(geometry.IsEmpty);
        }
        returnSqlBoolean.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBoolean IsClosed(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            if (geometry isILineString)
                returnnewSqlBoolean((geometry asILineString).IsClosed);
            if (geometry isMultiLineString)
                returnnewSqlBoolean((geometry asMultiLineString).IsClosed);
        }
        returnSqlBoolean.Null;
    }

```

```

        [SqlFunction(IsDeterministic = true)]
        public static SqlString GeometryType(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return new SqlString(geometry.GeometryType);
            }
            return SqlString.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes InteriorRingN(SqlBytes ewkb, SqlInt32 index)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                try
                {
                    {
                        if (geometry is IPolygon)
                        {
                            return ConvertGeometry.ToSqlBytes((geometry as IPolygon).GetInteriorRingN(index.Value - 1));
                        }
                    }
                }
                catch { }
            }
            return SqlBytes.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes GeometryN(SqlBytes ewkb, SqlInt32 index)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry is IGeometryCollection)
            {
                {
                    try
                    {
                        {
                            return ConvertGeometry.ToSqlBytes((geometry as IGeometryCollection).Geometries[index.Value - 1]);
                        }
                    }
                    catch
                    {
                        {
                            return SqlBytes.Null;
                        }
                    }
                }
            }
            else
            {
                return ewkb;
            }
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes ExteriorRing(SqlBytes ewkb)
        {
            IPolygon geometry = ConvertGeometry.ToGeometry(ewkb) as IPolygon;
            if (geometry != null)
            {
                {
                    return ConvertGeometry.ToSqlBytes(geometry.ExteriorRing);
                }
            }
            return SqlBytes.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes Envelope(SqlBytes ewkb)

```

```

    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            return ConvertGeometry.ToSqlBytes(geometry.Envelope);
        }
        return SqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    public static SqlBytes EndPoint(SqlBytes ewkb)
    {
        ILineString geometry = ConvertGeometry.ToGeometry(ewkb) as ILineString;
        if (geometry != null)
        {
            return ConvertGeometry.ToSqlBytes(geometry.EndPoint);
        }
        return SqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = false, DataAccess = DataAccessKind.Read)]
    public static SqlBytes EnvelopeQuery(
        SqlString table_name,
        SqlString column_name)
    {
        return EnvelopeQueryWhere(table_name, column_name, SqlString.Null);
    }

    [SqlFunction(IsDeterministic = false, DataAccess = DataAccessKind.Read)]
    public static SqlBytes EnvelopeQueryWhere(
        SqlString table_name,
        SqlString column_name,
        SqlString condition)
    {
        using (SqlConnection connection = Helper.ContextConnection)
        {
            connection.Open();
            using (SqlCommand command = connection.CreateCommand())
            {
                string tableName = table_name.Value;
                string columnName = column_name.Value;
                string schemaName = "";

                Helper.SplitSchemaAndTableName(command, ref tableName, ref schemaName);
                int srid = GeometryColumnsTable.GetSRID(command, schemaName, tableName, columnName);

                string filterQuery = "";
                if (!condition.IsNull && condition.Value.Trim().Length != 0)
                    filterQuery = "WHERE " + condition.Value;

                Helper.BuildCommand(command, string.Format(
@"SELECT
    Min([2]_Envelope_MinX)
    , Max([2]_Envelope_MaxX)
    , Min([2]_Envelope_MinY)
    , Max([2]_Envelope_MaxY)
FROM [{0}].[{1}]
{3}
"
                    , schemaName

```

```

        , tableName
        , columnName
        , filterQuery
    ));
using (SqlDataReader reader = command.ExecuteReader())
{
    if (reader.Read())
    {
        if (reader.IsDBNull(0))
        {
            returnSqlBytes.Null;
        }
        else
        {
            IGeometry envelope = GeometryFactory.Default.ToGeometry(
            newEnvelope(
                reader.GetDouble(0)
                , reader.GetDouble(1)
                , reader.GetDouble(2)
                , reader.GetDouble(3)
            ));
            returnConvertGeometry.ToSqlBytes(envelope, srid);
        }
    }
}
returnSqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstaticSqlBytes Expand(SqlBytes ewkb, SqlDouble distance)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null && !distance.IsNull)
    {
        IEnvelope envelope = geometry.EnvelopeInternal;
        envelope.ExpandBy(distance.Value);
        returnConvertGeometry.ToSqlBytes(Geometry.DefaultFactory.ToGeometry(envelope),
        geometry.SRID);
    }
    returnSqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstaticSqlBoolean IsValid(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        returnnewSqlBoolean(geometry.IsValid);
    }
    returnSqlBoolean.Null;
}

[SqlFunction(IsDeterministic = false, DataAccess = DataAccessKind.Read)]
publicstaticSqlBoolean IsValidSRID(SqlInt32 srid)
{
    returnnewSqlBoolean(SpatialExtLibrary.SpatialRefSysTable.IsValidSRID(srid.Value));
}

```

```

        [SqlFunction(IsDeterministic = true)]
        public static SqlDouble MaxX(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return new SqlDouble(geometry.EnvelopeInternal.MaxX);
            }
            return SqlDouble.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlDouble MaxY(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return new SqlDouble(geometry.EnvelopeInternal.MaxY);
            }
            return SqlDouble.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlDouble MinX(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return new SqlDouble(geometry.EnvelopeInternal.MinX);
            }
            return SqlDouble.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlDouble MinY(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return new SqlDouble(geometry.EnvelopeInternal.MinY);
            }
            return SqlDouble.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlInt32 NPoints(SqlBytes ewkb)
        {
            IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
            if (geometry != null)
            {
                return geometry.NumPoints;
            }
            return SqlInt32.Null;
        }
    #endregion

    #region GeoCalculations
    [SqlFunction(IsDeterministic = true)]
    public static SqlDouble Area(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
    
```

```

        {
returnnewSqlDouble(geometry.Area);
        }
returnSqlDouble.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBytes Boundary(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnConvertGeometry.ToSqlBytes(geometry.Boundary);
        }
        returnSqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBytes Buffer(SqlBytes ewkb, SqlDouble distance)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null&& !distance.IsNull)
        {
            returnConvertGeometry.ToSqlBytes(geometry.Buffer(distance.Value));
        }
        returnSqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBytes Centroid(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnConvertGeometry.ToSqlBytes(geometry.Centroid);
        }
        returnSqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBytes ConvexHull(SqlBytes ewkb)
    {
        IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
        if (geometry != null)
        {
            returnConvertGeometry.ToSqlBytes(geometry.ConvexHull());
        }
        returnSqlBytes.Null;
    }

    [SqlFunction(IsDeterministic = true)]
    publicstaticSqlBytes Difference(SqlBytes ewkb1, SqlBytes ewkb2)
    {
        IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
        if (geometry1 != null)
        {
            IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
            if (geometry2 != null)
            {
                if (geometry1.SRID != geometry2.SRID)
                thrownewDifferentSRIDException(geometry1.SRID, geometry2.SRID);
            }
        }
    }

```

```

        return ConvertGeometry.ToSqlBytes(geometry1.Difference(geometry2));
    }
}
return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBytes GeomUnion(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            {
                throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            }
            return ConvertGeometry.ToSqlBytes(geometry1.Union(geometry2));
        }
    }
    return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBytes Intersection(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            {
                throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            }
            return ConvertGeometry.ToSqlBytes(geometry1.Intersection(geometry2));
        }
    }
    return SqlBytes.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlDouble Length(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        return new SqlDouble(geometry.Length);
    }
    return SqlDouble.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBytes PointOnSurface(SqlBytes ewkb)
{
    IGeometry geometry = ConvertGeometry.ToGeometry(ewkb);
    if (geometry != null)
    {
        return ConvertGeometry.ToSqlBytes(geometry.InteriorPoint);
    }
    return SqlBytes.Null;
}

```

```

        [SqlFunction(IsDeterministic = true)]
        public static SqlBytes SymDifference(SqlBytes ewkb1, SqlBytes ewkb2)
        {
            IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
            if (geometry1 != null)
            {
                IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
                if (geometry2 != null)
                {
                    if (geometry1.SRID != geometry2.SRID)
                        throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
                    return ConvertGeometry.ToSqlBytes(geometry1.SymmetricDifference(geometry2));
                }
            }
            return SqlBytes.Null;
        }
        #endregion

        #region Relations
        [SqlFunction(IsDeterministic = true)]
        public static SqlBoolean Contains(SqlBytes ewkb1, SqlBytes ewkb2)
        {
            IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
            if (geometry1 != null)
            {
                IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
                if (geometry2 != null)
                {
                    if (geometry1.SRID != geometry2.SRID)
                        throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
                    return new SqlBoolean(geometry1.Contains(geometry2));
                }
            }
            return SqlBoolean.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBoolean CoveredBy(SqlBytes ewkb1, SqlBytes ewkb2)
        {
            IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
            if (geometry1 != null)
            {
                IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
                if (geometry2 != null)
                {
                    if (geometry1.SRID != geometry2.SRID)
                        throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
                    return new SqlBoolean(geometry2.Covers(geometry1));
                }
            }
            return SqlBoolean.Null;
        }

        [SqlFunction(IsDeterministic = true)]
        public static SqlBoolean Crosses(SqlBytes ewkb1, SqlBytes ewkb2)
        {
            IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
            if (geometry1 != null)
            {
                IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);

```

```

        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            thrownewDifferentSRIDException(geometry1.SRID, geometry2.SRID);
            returnnewSqlBoolean(geometry1.Crosses(geometry2));
        }
    }
    returnSqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstaticSqlDouble Distance(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            thrownewDifferentSRIDException(geometry1.SRID, geometry2.SRID);
            returnnewSqlDouble(geometry1.Distance(geometry2));
        }
    }
    returnSqlDouble.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstaticSqlBoolean Equals(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            thrownewDifferentSRIDException(geometry1.SRID, geometry2.SRID);
            returnnewSqlBoolean(geometry1.Equals(geometry2));
        }
    }
    returnSqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
publicstaticSqlBoolean EqualsExact(SqlBytes ewkb1, SqlBytes ewkb2, [DefaultValue(0)]
SqlDouble tolerance)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            thrownewDifferentSRIDException(geometry1.SRID, geometry2.SRID);
            if (tolerance.IsNull)
            returnnewSqlBoolean(geometry1.EqualsExact(geometry2));
            else
            returnnewSqlBoolean(geometry1.EqualsExact(geometry2, tolerance.Value));
        }
    }
}

```

```

    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean Intersects(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
                throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.Intersects(geometry2));
        }
    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean IsWithinDistance(SqlBytes ewkb1, SqlBytes ewkb2, SqlDouble
distance)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null && !distance.IsNull)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
                throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.IsWithinDistance(geometry2, distance.Value));
        }
    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean Overlaps(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
                throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.Overlaps(geometry2));
        }
    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean Relate(SqlBytes ewkb1, SqlBytes ewkb2, SqlString
intersectionPattern)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);

```

```

    if (geometry1 != null && !intersectionPattern.IsNull)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.Relate(geometry2, intersectionPattern.Value));
        }
    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlString RelatePattern(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlString(geometry1.Relate(geometry2).ToString());
        }
    }
    return SqlString.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean Touches(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.Touches(geometry2));
        }
    }
    return SqlBoolean.Null;
}

[SqlFunction(IsDeterministic = true)]
public static SqlBoolean Within(SqlBytes ewkb1, SqlBytes ewkb2)
{
    IGeometry geometry1 = ConvertGeometry.ToGeometry(ewkb1);
    if (geometry1 != null)
    {
        IGeometry geometry2 = ConvertGeometry.ToGeometry(ewkb2);
        if (geometry2 != null)
        {
            if (geometry1.SRID != geometry2.SRID)
            throw new DifferentSRIDException(geometry1.SRID, geometry2.SRID);
            return new SqlBoolean(geometry1.Within(geometry2));
        }
    }
}

```

```

returnSqlBoolean.Null;
    }
    #endregion
};

```

Exceptions.cs

```

namespace SpatialExtLibrary
{
    publicclassInvalidSRIDException : ApplicationException
    {
        public InvalidSRIDException(int srid)
            : base(string.Format("Invalid SRID {0}. Valid values are -1 and
those in [ST].[SPATIAL_REF_SYS] table.", srid)) { }

        publicclassDifferentSRIDException : ApplicationException
        {
            public DifferentSRIDException(int srid1, int srid2)
                : base(string.Format("Operation on geometries with different SRIDs
({0} and {1})", srid1, srid2)) { }
        }
    }
}

```

GeometryTransform.cs

```

namespace GisSharpBlog.NetTopologySuite.CoordinateSystems.Transformations
{
    publicclassGeometryTransform
    {
        privatestaticIPoint ToNTS(double x, double y)
        {
            returnnewPoint(x, y);
        }

        privatestaticdouble[] ToLightStruct(double x, double y)
        {
            returnnewdouble[] { x, y, };
        }

        publicstaticIEnvelope TransformBox(IEnvelope box, IMathTransform
transform)
        {
            if (box == null)
                returnnull;
            double[][] corners = newdouble[4][];
            corners[0] = transform.Transform(ToLightStruct(box.MinX, box.MinY)); //LL
            corners[1] = transform.Transform(ToLightStruct(box.MaxX, box.MaxY)); //UR
            corners[2] = transform.Transform(ToLightStruct(box.MinX, box.MaxY)); //UL
            corners[3] = transform.Transform(ToLightStruct(box.MaxX, box.MinY)); //LR

            IEnvelope result = newEnvelope();
            foreach (double[] p in corners)
                result.ExpandToInclude(p[0], p[1]);
            return result;
        }
    }
}

```

```

transform)    public static IGeometry TransformGeometry(IGeometry g, IMathTransform
              {
                if (g == null)
                    return null;
                elseif (g is IPoint)
                    return TransformPoint(g as IPoint, transform);
                elseif (g is ILineString)
                    return TransformLineString(g as ILineString, transform);
                elseif (g is IPolygon)
                    return TransformPolygon(g as IPolygon, transform);
                elseif (g is IMultiPoint)
                    return TransformMultiPoint(g as IMultiPoint, transform);
                elseif (g is IMultiLineString)
                    return TransformMultiLineString(g as IMultiLineString,
transform);
                elseif (g is IMultiPolygon)
                    return TransformMultiPolygon(g as IMultiPolygon, transform);
                else throw new ArgumentException("Could not transform geometry type
'" + g.GetType().ToString() + "'");
            }

            public static IPoint TransformPoint(IPoint p, IMathTransform transform)
            {
                try
                {
double[] point = transform.Transform(ToLightStruct(p.X, p.Y));
return ToNTS(point[0], point[1]);
                }
                catch { return null; }
            }

            public static ILineString TransformLineString(ILineString l, IMathTransform
transform)
            {
                try
                {
transform);
                    List<ICoordinate> coords = ExtractCoordinates(l,
                    return new LineString(coords.ToArray());
                }
                catch { return null; }
            }

            public static ILinearRing TransformLinearRing(ILinearRing r, IMathTransform
transform)
            {
                try
                {
List<ICoordinate> coords = ExtractCoordinates(r, transform);
return new LinearRing(coords.ToArray());
                }
                catch { return null; }
            }

            private static List<ICoordinate> ExtractCoordinates(ILineString ls, IMathTransform
transform)
            {
List<double[]> points =

```

```

newList<double[]>(ls.NumPoints);
foreach (ICoordinate c in ls.Coordinates)
    points.Add(ToLightStruct(c.X, c.Y));
    points = transform.TransformList(points);
List<ICoordinate> coords = newList<ICoordinate>(points.Count);
foreach (double[] p in points)
    coords.Add(newCoordinate(p[0], p[1]));
return coords;
}

    publicstaticIPolygon TransformPolygon(IPolygon p, IMathTransform
transform)
    {
        List<ILinearRing> rings =
newList<ILinearRing>(p.InteriorRings.Length);
for (int i = 0; i < p.InteriorRings.Length; i++)

        rings.Add(TransformLinearRing((ILinearRing)p.InteriorRings[i], transform));
        returnnewPolygon(TransformLinearRing((ILinearRing)p.ExteriorRing,
transform), rings.ToArray());
    }

    publicstaticIMultiPoint TransformMultiPoint(IMultiPoint points,
IMathTransform transform)
    {
List<double[]> pointList = newList<double[]>(points.Geometries.Length);
        foreach (IPoint p in points.Geometries)
            pointList.Add(ToLightStruct(p.X, p.Y));
            pointList = transform.TransformList(pointList);
            IPoint[] array = newIPoint(pointList.Count);
for (int i = 0; i < pointList.Count; i++)
            array[i] = ToNTS(pointList[i][0], pointList[i][1]);
            returnnewMultiPoint(array);
    }

    publicstaticIMultiLineString TransformMultiLineString(IMultiLineString
lines, IMathTransform transform)
    {
        List<ILineString> strings =
newList<ILineString>(lines.Geometries.Length);
        foreach (ILineString ls in lines.Geometries)
            strings.Add(TransformLineString(ls, transform));
returnnewMultiLineString(strings.ToArray());
    }

    publicstaticIMultiPolygon TransformMultiPolygon(IMultiPolygon polys,
IMathTransform transform)
    {
        List<IPolygon> polygons =
newList<IPolygon>(polys.Geometries.Length);
        foreach (IPolygon p in polys.Geometries)
            polygons.Add(TransformPolygon(p, transform));
        returnnewMultiPolygon(polygons.ToArray());
    }

    publicstaticIGeometryCollection
TransformGeometryCollection(GeometryCollection geoms, IMathTransform transform)
    {
        List<IGeometry> coll =
newList<IGeometry>(geoms.Geometries.Length);
        foreach (IGeometry g in geoms.Geometries)

```

```

        coll.Add(TransformGeometry(g, transform));
        return new GeometryCollection(coll.ToArray());
    }
}

```

Helper.cs

```

namespace SpatialExtLibrary
{
    public static class Helper
    {
        private static readonly string[] contextConnection = new string[1];

        public static SqlConnection ContextConnection
        {
            get
            {
                return new SqlConnection(ContextConnectionString);
            }
        }

        public static string ContextConnectionString
        {
            get
            {
                if (SqlContext.IsAvailable)
                    return "context connection=true";
                else
                    return contextConnection[0];
            }
            set
            {
                contextConnection[0] = value;
            }
        }

        internal static bool CheckQueryExists(SqlCommand command, string query,
        params object[] parameters)
        {
            try
            {
                BuildCommandArray(command, query, parameters);
                object ret = command.ExecuteScalar();
                return (ret != null);
            }
            catch
            {
                return false;
            }
        }

        internal static bool CheckMetaTableExists(SqlCommand command, string
        tableName)
        {
            return CheckQueryExists(command, string.Format(
                @"SELECT [name] FROM [sys].[objects] WHERE [object_id] =
                OBJECT_ID(N'[ST].[{0}]') AND [type] = N'U'"
                , tableName));
        }
    }
}

```

```

        internalstaticbool CheckTriggerExists(SqlCommand command, string
schemaName, string triggerName)
        {
            return CheckQueryExists(command,
@"SELECT t.object_id
        FROM sys.triggers t, sys.objects o, sys.schemas s
        WHERE t.object_id = o.object_id
        AND o.schema_id = s.schema_id
        AND s.name = @p0
        AND t.name = @p1"
        , schemaName
        , triggerName
        );
        }

        internalstaticbool CheckColumnExists(SqlCommand command, string
schemaName, string tableName, string columnName)
        {
            return CheckQueryExists(command, string.Format(
@"SELECT object_id
        FROM sys.columns
        WHERE object_id = OBJECT_ID(N'[{0}].[{1}]')
        AND name = @p0"
        , schemaName
        , tableName
        )
        , columnName
        );
        }

        internalstaticvoid DropColumn(SqlCommand command, string schemaName,
string tableName, string columnName)
        {
            if (Helper.CheckColumnExists(command, schemaName, tableName,
columnName))
            {
                Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
        DROP COLUMN [{2}]
"
        , schemaName, tableName, columnName));
                command.ExecuteNonQuery();
            }
        }

        internalstaticvoid DropIndex(SqlCommand command, string schemaName,
string tableName, string indexName)
        {
            if (Helper.CheckIndexExists(command, schemaName, tableName,
indexName))
            {
                Helper.BuildCommand(command, string.Format(
@"DROP INDEX [{2}]
        ON [{0}].[{1}]
"
        , schemaName, tableName, indexName));
                command.ExecuteNonQuery();
            }
        }

```

```

        internalstaticbool CheckIndexExists(SqlCommand command, string
schemaName, string tableName, string indexName)
        {
            return CheckQueryExists(command, string.Format(
@"
                SELECT object_id
                FROM sys.indexes
                WHERE object_id = OBJECT_ID(N'[{0}].[{1}]')
                AND name = @p0"
                , schemaName
                , tableName
                )
                , indexName
                );
        }

        internalstaticbool CheckConstraintExists(SqlCommand command, string
schemaName, string tableName, string constraintName)
        {
            return CheckQueryExists(command, string.Format(
@"SELECT object_id
FROM sys.check_constraints
WHERE object_id = OBJECT_ID(N'[{0}].[{2}]')
AND parent_object_id = OBJECT_ID(N'[{0}].[{1}]')"
                , schemaName
                , tableName
                , constraintName
                ));
        }

        internalstaticstring CheckSchemaName(SqlCommand command, SqlString
schemaName)
        {
            return CheckSchemaName(command, schemaName.ToString());
        }

        internalstaticstring CheckSchemaName(SqlCommand command, string
schemaName)
        {
            string checkedSchema = "dbo";
            if (schemaName.Trim().Length != 0)
            {
                bool schemaExists = CheckQueryExists(command,
@"SELECT [name] FROM [sys].[schemas] WHERE [name] = @p0"
                , schemaName);

                if (schemaExists)
                {
                    checkedSchema = schemaName;
                }
            }
            return checkedSchema;
        }

        internalstaticvoid ExecuteDropTable(SqlCommand command, string
schemaName, string tableName)
        {
            Helper.BuildCommand(command, string.Format(
@"DROP TABLE [{0}].[{1}]"
                , schemaName, tableName));
        }

```

```

        command.ExecuteNonQuery();
    }

    internal static void SplitSchemaAndTableName(SqlCommand command, ref string
tableName, ref string schemaName)
    {
        int schemaDot = tableName.IndexOf(".");
        if (schemaDot != -1)
        {
            schemaName = tableName.Substring(0, schemaDot);
            tableName = tableName.Substring(schemaDot + 1);
        }
        schemaName = CheckSchemaName(command, schemaName);
    }

    internal static void BuildCommand(SqlCommand command, string commandText,
params object[] parameters)
    {
        BuildCommandArray(command, commandText, parameters);
    }

    internal static void BuildCommandArray(SqlCommand command, string
commandText, object[] parameters)
    {
        command.CommandText = commandText;
        command.CommandType = CommandType.Text;
        command.Parameters.Clear();
        for (int i = 0; i < parameters.Length; i++)
        {
            command.Parameters.AddWithValue("@p" + i.ToString(),
parameters[i]);
        }
    }
}
}

```

SPs.cs

```

public sealed partial class SPs
{
    [SqlProcedure]
    public static void AddGeometryColumn(
        [DefaultValue("")] SqlString schema_name,
        SqlString table_name,
        SqlString column_name,
        [DefaultValue(-1)] SqlInt32 srid,
        [DefaultValue("GEOMETRY")] SqlString type)
    {
        string tableName = table_name.Value;
        string columnName = column_name.Value;
        int sridValue = srid.Value;
        string typeValue = type.Value;

        if (!IsValidGeometryType(typeValue))
        {
            throw new ApplicationException(
@"Invalid geometry type. Valid ones are:
        \n\t" + string.Join("\n\t", validGeometryTypes));
        }
    }
}

```

```

    }

    using (SqlConnection connection = Helper.ContextConnection)
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();
        try
        {
            using (SqlCommand command = connection.CreateCommand())
            {
                command.Transaction = transaction;

                ExecuteCreateMetaTables(command);

                if (!SpatialRefSysTable.CheckSRIDExists(command, sridValue))
                {
                    throw new InvalidSRIDException(sridValue);
                }

                string schemaName = Helper.CheckSchemaName(command, schema_name);

                GeometryColumnsTable.ExecuteAddMetaRecord(command, schemaName, tableName, columnName,
                    sridValue, typeValue);
                GeometryTable.ExecuteAddColumn(command, schemaName, tableName, columnName);
                GeometryTable.ExecuteAddIndexes(command, schemaName, tableName, columnName);
                GeometryTable.ExecuteAddConstraintSRID(command, schemaName, tableName, columnName,
                    sridValue);
                GeometryTable.ExecuteAddConstraintGeometryType(command, schemaName, tableName,
                    columnName, typeValue);
                GeometryTable.ExecuteCreateTrigger(command, schemaName, tableName, columnName);
                ExecuteAddGeometryQueryFunctions(command, schemaName, tableName,
                    columnName);

                // If you add more associated objects to the column, please
                // verify whether they have to be added to "ImportFromDatasource" method as well.

                transaction.Commit();
                SqlContext.Pipe.Send(string.Format(
                    @"Added geometry column [{0}].[{1}].[{2}] and constraints SRID={3}, GeometryType={4}."
                    , schemaName, tableName, columnName, sridValue, typeValue));
            }
        }
        catch
        {
            transaction.Rollback();
        }
        throw;
    }
}

private static readonly string[] validGeometryTypes = new string[] {
    "GEOMETRY",
    "GEOMETRYCOLLECTION",
    "POINT",
    "MULTIPOINT",
    "POLYGON",
    "MULTIPOLYGON",
    "LINESTRING",
    "MULTILINESTRING"};

```

```

private static bool IsValidGeometryType(string type)
{
    return new List<string>(validGeometryTypes).Contains(type.ToUpperInvariant());
}

public static void ExecuteCreateMetaTables(SqlCommand command)
{
    GeometryColumnsTable.ExecuteCreate(command);
    SpatialRefSysTable.ExecuteCreate(command);
}

public static void ExecuteBackupMetaTables(SqlCommand command)
{
    GeometryColumnsTable.ExecuteBackup(command);
    SpatialRefSysTable.ExecuteBackup(command);
}

public static void ExecuteAddGeometryQueryFunctions(SqlCommand command, string
schemaName, string tableName, string columnName)
{
    Helper.BuildCommand(command, "[ST].[AddGeometryQueryFunctions]");
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.AddWithValue("@schema_name", schemaName);
    command.Parameters.AddWithValue("@table_name", tableName);
    command.Parameters.AddWithValue("@column_name", columnName);
    command.ExecuteNonQuery();
}

public static void ExecuteDropGeometryQueryFunctions(SqlCommand command, string
schemaName, string tableName, string columnName)
{
    Helper.BuildCommand(command, "[ST].[DropGeometryQueryFunctions]");
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.AddWithValue("@schema_name", schemaName);
    command.Parameters.AddWithValue("@table_name", tableName);
    command.Parameters.AddWithValue("@column_name", columnName);
    command.ExecuteNonQuery();
}

[SqlProcedure]
public static void DropGeometryColumn(
    SqlString schema_name,
    SqlString table_name,
    SqlString column_name)
{
    using (SqlConnection connection = Helper.ContextConnection)
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();
        try
        {
            using (SqlCommand command = connection.CreateCommand())
            {
                command.Transaction = transaction;
                string schemaName = Helper.CheckSchemaName(command, schema_name);
                string tableName = table_name.Value;
                string columnName = column_name.Value;

                if (!GeometryColumnsTable.CheckGeometryColumnExists(command, schemaName, tableName,
                    columnName))

```

```

        {
thrownewApplicationException("Column not found in [ST].[GEOMETRY_COLUMNS] table.");
        }

GeometryTable.ExecuteDropColumnDependantObjects(command, schemaName, tableName,
columnName);
Helper.DropColumn(command, schemaName, tableName, columnName);

        transaction.Commit();
SqlContext.Pipe.Send(string.Format(
@"Column [{0}].[{1}].[{2}] removed."
        , schemaName, tableName, columnName));
    }
}
catch
{
    transaction.Rollback();
throw;
}
}

[SqlProcedure]
publicstaticvoid ImportFromShapefile(
SqlString shapefile, [DefaultValue("")] SqlString schema_name,
[DefaultValue("")] SqlString table_name, [DefaultValue("")] SqlString
column_name,
[DefaultValue(-1)] SqlInt32 srid, [DefaultValue("GEOMETRY")] SqlString type,
[DefaultValue(false)] SqlBoolean appendRows
)
{
long importedRows;
TimeSpan elapsedTime;

    ImportFromShapefileMain(shapefile.Value, schema_name.Value, table_name.Value,
        column_name.Value, srid.Value, type.Value, "context connection=true",
        appendRows.IsTrue, out importedRows, out elapsedTime);

SqlContext.Pipe.Send(string.Format("{0} records imported in {1}.", importedRows,
FormatElapsedTime(elapsedTime)));

    }

publicstaticstring FormatElapsedTime(TimeSpan elapsedTime)
{
    if (elapsedTime.Minutes < 1)
returnstring.Format("{0} seconds", elapsedTime.Seconds);
elseif (elapsedTime.Hours < 1)
returnstring.Format("{0} minutes", elapsedTime.TotalMinutes.ToString("N"));
else
return elapsedTime.ToString();
    }

publicstaticvoid ImportFromShapefileMain(
string shapefile, string schema_name, string table_name, string column_name,
int srid, string type, string connectionString, bool appendRows, outlong importedRows,
outTimeSpan elapsedTime
)
{
    string filename = Path.GetFileName(shapefile);

```

```

if (string.IsNullOrEmpty(table_name))
    table_name = filename.Remove(filename.Length - 4, 4);

if (string.IsNullOrEmpty(column_name))
    column_name = "the_geom";

using (IDataImportProcess process = new ShapefileImportProcess(shapefile))
    DataImportProcess(process, schema_name, table_name, column_name, srid,
type, connectionString, appendRows, out importedRows, out elapsedTime);

    }

public interface IDataImportProcess : IDisposable
{
    Dictionary<string, Type> GetColumns();
    void Execute(SqlCommand command, string columnName, int srid, out long importedRows);
}

public class ShapefileImportProcess : IDataImportProcess
{
    ShapefileDataReader reader;
    Dictionary<string, Type> columns;

    public ShapefileImportProcess(string filename)
    {
        reader = new ShapefileDataReader(filename, GeometryFactory.Default);
    }

    public Dictionary<string, Type> GetColumns()
    {
        if (columns == null)
        {
            columns = new Dictionary<string, Type>();
            foreach (DbaseFieldDescriptor col in reader.DbaseHeader.Fields)
            {
                columns[col.Name] = col.Type;
            }
        }
        return columns;
    }

    public void Execute(SqlCommand command, string columnName, int srid, out long
importedRows)
    {
        importedRows = 0;
        Dictionary<string, Type> columns = GetColumns();

        while (reader.Read())
        {
            int i = 0;
            foreach (string column in columns.Keys)
            {
                command.Parameters["@" + column].Value = reader.GetValue(i);
                i++;
            }
            command.Parameters["@" + columnName].Value =
ConvertGeometry.ToSqlBytes(reader.Geometry, srid).Value;

            command.ExecuteNonQuery();
            importedRows++;
        }
    }
}

```

```

    }

    public void Dispose()
    {
        reader.Dispose();
    }
}

public static void DataImportProcess(
    IDataImportProcess process,
    string schemaName,
    string tableName,
    string columnName,
    int srid,
    string type,
    string connectionString,
    bool appendRows,
    out long importedRows,
    out TimeSpan elapsedTime
)
{
    importedRows = -1;
    elapsedTime = TimeSpan.Zero;

    DateTime start = DateTime.Now;

    Helper.ContextConnectionString = connectionString;

    if (!SpatialRefSysTable.IsValidSRID(srid))
    {
        throw new InvalidSRIDException(srid);
    }

    using (SqlConnection connection = Helper.ContextConnection)
    {
        connection.Open();
        using (SqlTransaction transaction = connection.BeginTransaction())
        {
            using (SqlCommand command = connection.CreateCommand())
            {
                command.Transaction = transaction;
                command.CommandTimeout = 0;
                ExecuteCreateMetaTables(command);
                schemaName = Helper.CheckSchemaName(command, schemaName);
                Dictionary<string, Type> columns = process.GetColumns();

                if (appendRows)
                {
                    GeometryTable.ExecuteDropIndexes(command, schemaName, tableName, columnName);
                    GeometryTable.ExecuteDropConstraintSRID(command, schemaName, tableName, columnName);
                    GeometryTable.ExecuteDropConstraintGeometryType(command, schemaName, tableName,
                        columnName);
                    GeometryTable.ExecuteDropTrigger(command, schemaName, tableName, columnName);
                }
                else
                {
                    try
                    {
                        Helper.ExecuteDropTable(command, schemaName, tableName);
                    }
                    catch { }
                }
            }
        }
    }
}

```

```

string createColumns = "";
foreach (KeyValuePair<string, Type> col in columns)
{
    if (col.Value != typeof(String))
        createColumns += ", " + col.Key + " " +
            Type2SqlType(col.Value).ToString();
    else
        createColumns += ", " + col.Key + " VARCHAR(4000)";
}
GeometryTable.ExecuteCreate(command, schemaName, tableName, columnName,
createColumns);
}

CreateInsertCommand(command, schemaName, tableName, columnName,
columns);
process.Execute(command, columnName, srid, out importedRows);
command.Parameters.Clear();

if (!appendRows)
{
    GeometryColumnsTable.ExecuteAddMetaRecord(command, schemaName, tableName, columnName,
srid, type);
    ExecuteAddGeometryQueryFunctions(command, schemaName,
tableName, columnName);
}
GeometryTable.ExecuteAddIndexes(command, schemaName, tableName, columnName);
GeometryTable.ExecuteAddConstraintSRID(command, schemaName, tableName, columnName,
srid);
GeometryTable.ExecuteAddConstraintGeometryType(command, schemaName, tableName,
columnName, type);
GeometryTable.ExecuteCreateTrigger(command, schemaName, tableName, columnName);

    transaction.Commit();
}
}

    elapsedTime = DateTime.Now - start;
}

private static void CreateInsertCommand(SqlCommand command, string schemaName, string
tableName, string geometryColumnName, Dictionary<string, Type> columns)
{
    string strSQL = " (";
    foreach (KeyValuePair<string, Type> col in columns)
        strSQL += "@" + col.Key + ",";

    strSQL += "@" + geometryColumnName + ")";
    strSQL = "INSERT INTO [" + schemaName + "].[" + tableName + "] " +
strSQL.Replace("@", "") + " VALUES " + strSQL;

    Helper.BuildCommand(command, strSQL);

    foreach (KeyValuePair<string, Type> col in columns)
    if (col.Value == typeof(string))
        command.Parameters.Add(@" + col.Key, SqlDbType.VarChar, 4000);
    else
        command.Parameters.Add(@" + col.Key, Type2SqlType(col.Value));

```

```

        command.Parameters.Add("@ " + geometryColumnName, SqlDbType.VarBinary,
int.MaxValue);

        command.Prepare();
    }

private static SqlDbType Type2SqlType(Type t)
{
    switch (t.ToString())
    {
        case "System.Boolean": return System.Data.SqlDbType.Bit;
        case "System.Single": return System.Data.SqlDbType.Real;
        case "System.Double": return System.Data.SqlDbType.Float;
        case "System.Int16": return System.Data.SqlDbType.SmallInt;
        case "System.Int32": return System.Data.SqlDbType.Int;
        case "System.Int64": return System.Data.SqlDbType.BigInt;
        case "System.DateTime": return System.Data.SqlDbType.DateTime;
        case "System.Byte[]": return System.Data.SqlDbType.Image;
        case "System.String": return System.Data.SqlDbType.VarChar;
        default:
            throw (new NotSupportedException("Unsupported datatype '" + t.Name + "' found in
datasource"));
    }
}

[SqlProcedure]
public static void UpdateGeometrySRID(
    SqlString schema_name,
    SqlString table_name,
    SqlString column_name,
    SqlInt32 srid)
{
    using (SqlConnection connection = Helper.ContextConnection)
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();
        try
        {
            using (SqlCommand command = connection.CreateCommand())
            {
                command.Transaction = transaction;
                string schemaName = Helper.CheckSchemaName(command, schema_name);
                string tableName = table_name.Value;
                string columnName = column_name.Value;
                int sridValue = srid.Value;

                if (!SpatialRefSysTable.CheckSRIDExists(command, sridValue))
                {
                    throw new InvalidSRIDException(sridValue);
                }

                if (!GeometryColumnsTable.CheckGeometryColumnExists(command, schemaName, tableName,
columnName))
                {
                    throw new ApplicationException("Column not found in [ST].[GEOMETRY_COLUMNS] table.");
                }

                Helper.BuildCommand(command, string.Format(
@"UPDATE [ST].[GEOMETRY_COLUMNS]
    SET [SRID] = {0}
    WHERE [F_TABLE_SCHEMA] = @p0

```

```

        AND [F_TABLE_NAME] = @p1
        AND [F_GEOMETRY_COLUMN] = @p2"
        , sridValue)
        , schemaName
        , tableName
        , columnName
    );
    command.ExecuteNonQuery();

    Helper.BuildCommand(command, string.Format(
@"ALTER TABLE [{0}].[{1}]
    DROP CONSTRAINT [enforce_srid_{2}]"
        , schemaName, tableName, columnName));
    command.ExecuteNonQuery();

    Helper.BuildCommand(command, string.Format(
@"UPDATE [{0}].[{1}]
    SET [{2}] = [ST].[SetSRID]([{2}], {3})"
        , schemaName, tableName, columnName, sridValue));
    command.ExecuteNonQuery();

    GeometryTable.ExecuteAddConstraintSRID(command, schemaName, tableName, columnName,
sridValue);

        transaction.Commit();
    SqlContext.Pipe.Send(string.Format(
@"Column SRID [{0}].[{1}].[{2}] changed to {3}."
        , schemaName, tableName, columnName, sridValue));
    }
}
catch
{
    transaction.Rollback();
throw;
}
}
};

```

Triggers.cs

```

public sealed partial class Triggers
{
    [SqlTrigger(Name = "ST_DropGeometryTable", Target = "DATABASE", Event = "FOR
DROP_TABLE")]
    public static void DropGeometryTableTrigger()
    {
        XmlDocument eventData = new XmlDocument();
        eventData.Load(SqlContext.TriggerContext.EventData.CreateReader());
        string schemaName =
eventData.SelectSingleNode("/EVENT_INSTANCE/SchemaName").InnerText;
        string tableName = eventData.SelectSingleNode("/EVENT_INSTANCE/ObjectName").InnerText;

        using (SqlConnection connection = Helper.ContextConnection)
        {
            connection.Open();
            using (SqlCommand command = connection.CreateCommand())
            {

```

```

GeometryTable.ExecuteDropTableDependantObjects(command, schemaName, tableName);
    }
}

SqlContext.Pipe.Send(string.Format(
@"Geometry table [{0}].[{1}] dropped."
    , schemaName, tableName));
}
}

```

TVFs.cs

```

public partial class TVFs
{
    class SpatialQuery
    {
        internal struct Arguments
        {
            {
                public SqlDouble double1;
                public SqlString string1;

                public Arguments(SqlDouble _double1)
                {
                    double1 = _double1;
                    string1 = SqlString.Null;
                }

                public Arguments(SqlString _string1)
                {
                    string1 = _string1;
                    double1 = SqlDouble.Null;
                }
            }

            public static Arguments Empty
            {
                get
                {
                    {
                        Arguments args;
                        args.double1 = SqlDouble.Null;
                        args.string1 = SqlString.Null;
                    }
                    return args;
                }
            }

            internal delegate IEnumerable<ReaderHandler> (SqlDataReader reader, IGeometry geometry,
                Arguments args);

            internal static IEnumerable Execute(
                SqlString table_name,
                SqlString column_name,
                SqlBytes ewkbMBR,
                SpatialQuery.ReaderHandler handler)
            {
                return Execute(table_name, column_name, ewkbMBR, null, SqlString.Null,
                    Arguments.Empty, handler);
            }

            internal static IEnumerable Execute(
                SqlString table_name,

```

```

    SqlString column_name,
    SqlBytes ewkbMBR,
    SqlString condition,
    SpatialQuery.ReaderHandler handler)
    {
    return Execute(table_name, column_name, ewkbMBR, null, condition, Arguments.Empty,
    handler);
    }

    internalstaticIEnumerable Execute(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkbMBR,
    SpatialQuery.Arguments args,
    SpatialQuery.ReaderHandler handler)
    {
    return Execute(table_name, column_name, ewkbMBR, null, SqlString.Null, args, handler);
    }

    internalstaticIEnumerable Execute(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkbMBR,
    SqlString condition,
    SpatialQuery.Arguments args,
    SpatialQuery.ReaderHandler handler)
    {
    return Execute(table_name, column_name, ewkbMBR, null, condition, args, handler);
    }

    internalstaticIEnumerable Execute(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkbMBR,
    SqlBytes ewkb,
    SpatialQuery.Arguments args,
    SpatialQuery.ReaderHandler handler)
    {
    return Execute(table_name, column_name, ewkbMBR, ewkb, SqlString.Null, args, handler);
    }

    internalstaticIEnumerable Execute(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkbMBR,
    SqlBytes ewkb,
    SqlString condition,
    SpatialQuery.Arguments args,
    SpatialQuery.ReaderHandler handler)
    {
    if (ewkbMBR == null || ewkbMBR.IsNull)
        ewkbMBR = ewkb;

    IGeometry geometryMBR = ConvertGeometry.ToGeometry(ewkbMBR);

    IGeometry geometry;
    if (ewkb == null || ewkb.IsNull)
        geometry = geometryMBR;
    else
        geometry = ConvertGeometry.ToGeometry(ewkb);

```

```

using (SqlConnection connection = Helper.ContextConnection)
{
    connection.Open();
    using (SqlCommand command = connection.CreateCommand())
    {
        string tableName = table_name.Value;
        string columnName = column_name.Value;

        string schemaName = "";
        Helper.SplitSchemaAndTableName(command, ref tableName, ref schemaName);

        string primaryKey = GeometryTable.GetPrimaryKeyColumn(command, schemaName, tableName);

        string filterQuery = "";

        if (geometryMBR != null && !geometryMBR.IsEmpty)
        {
            if (!condition.IsNull && condition.Value.Trim().Length != 0)
                filterQuery = " AND (" + condition.Value + ")";
            IEnvelope mbr = geometryMBR.EnvelopeInternal;
            Helper.BuildCommand(command, string.Format(
@"SELECT [{4}], [{2}]
        FROM [{0}].[{1}] WITH (INDEX(IDX_{2}_Envelope))
        WHERE
        (    [{2}_Envelope_MinX] < @p0
        AND [{2}_Envelope_MaxX] > @p1
        AND [{2}_Envelope_MinY] < @p2
        AND [{2}_Envelope_MaxY] > @p3
        ){3}"
        , schemaName
        , tableName
        , columnName
        , filterQuery
        , primaryKey
        )
        , mbr.MaxX
        , mbr.MinX
        , mbr.MaxY
        , mbr.MinY
        ));
        }
        else
        {
            if (!condition.IsNull && condition.Value.Trim().Length != 0)
                filterQuery = "WHERE " + condition.Value;
            Helper.BuildCommand(command, string.Format(
@"SELECT [{4}], [{2}]
        FROM [{0}].[{1}]
        {3}"
        , schemaName
        , tableName
        , columnName
        , filterQuery
        , primaryKey
        ));
        }

        using (SqlDataReader reader = command.ExecuteReader())
        {
            return handler.Invoke(reader, geometry, args);
        }
    }
}

```

```

        }
    }
}

internalconststring TableDefinition = "oid bigint";
internalconststring FillRow = "FillRowSpatialQuery";

}

publicstaticvoid FillRowSpatialQuery(Object obj, outSqlInt64 oid)
{
    oid = newSqlInt64((Int64)obj);
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
publicstaticIEnumerable EqualsExactQuery(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb,
    SqlDouble tolerance)
{
    returnSpatialQuery.Execute(table_name, column_name, ewkb,
    newSpatialQuery.Arguments(tolerance), EqualsExactQueryHandler);
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
publicstaticIEnumerable EqualsExactQueryWhere(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb,
    SqlDouble tolerance,
    SqlString condition)
{
    returnSpatialQuery.Execute(table_name, column_name, ewkb, condition,
    newSpatialQuery.Arguments(tolerance), EqualsExactQueryHandler);
}

privatestaticIEnumerable EqualsExactQueryHandler(SqlDataReader reader, IGeometry
geometry, SpatialQuery.Arguments args)
{
    double tolerance;
    if (args.double1.IsNull)
        tolerance = 0;
    else
        tolerance = args.double1.Value;

    List<Int64> oids = newList<Int64>();
    while (reader.Read())

```

```

        {
            IGeometry _geometry = ConvertGeometry.ToGeometry(reader.GetSqlBytes(1));
            if (geometry.EqualsExact(_geometry, tolerance))
            {
                oids.Add((long)reader.GetValue(0));
            }
        }
    }
    return oids;
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
public static IEnumerable FilterQuery(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb)
{
    return SpatialQuery.Execute(table_name, column_name, ewkb, SqlString.Null,
        FilterQueryHandler);
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
public static IEnumerable FilterQueryWhere(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb,
    SqlString condition)
{
    return SpatialQuery.Execute(table_name, column_name, ewkb, condition,
        FilterQueryHandler);
}

private static IEnumerable FilterQueryHandler(SqlDataReader reader, IGeometry geometry,
    SpatialQuery.Arguments args)
{
    {
        List<Int64> oids = newList<Int64>();
        while (reader.Read())
        {
            oids.Add((long)reader.GetValue(0));
        }
    }
    return oids;
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = IsWithinDistanceQuery_FillRow
    , TableDefinition = IsWithinDistanceQuery_TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]

```

```

publicstaticIEnumerable IsWithinDistanceQuery(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb,
    SqlDouble distance)
{
    returnSpatialQuery.Execute(table_name, column_name, CustomFunctions.Expand(ewkb,
distance), ewkb, newSpatialQuery.Arguments(distance), IsWithinDistanceQueryHandler);
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = IsWithinDistanceQuery_FillRow
    , TableDefinition = IsWithinDistanceQuery_TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
publicstaticIEnumerable IsWithinDistanceQueryWhere(
    SqlString table_name,
    SqlString column_name,
    SqlBytes ewkb,
    SqlDouble distance,
    SqlString condition)
{
    returnSpatialQuery.Execute(table_name, column_name, CustomFunctions.Expand(ewkb,
distance), ewkb, condition, newSpatialQuery.Arguments(distance),
IsWithinDistanceQueryHandler);
}

internalconststring IsWithinDistanceQuery_TableDefinition = "oid bigint, distance
float";
internalconststring IsWithinDistanceQuery_FillRow = "FillRowIsWithinDistanceQuery";

publicstaticvoid FillRowIsWithinDistanceQuery(Object obj, outSqlInt64 oid,
outSqlDouble distance)
{
    oid = newSqlInt64((Int64)((object[])obj)[0]);
    distance = newSqlDouble((double)((object[])obj)[1]);
}

privatestaticIEnumerable IsWithinDistanceQueryHandler(SqlDataReader reader, IGeometry
geometry, SpatialQuery.Arguments args)
{
    double distance = args.double1.Value;
    List<object[]> rows = newList<object[]>();
    while (reader.Read())
    {
        IGeometry _geometry = ConvertGeometry.ToGeometry(reader.GetSqlBytes(1));

        double currentDistance = geometry.Distance(_geometry);
        if (currentDistance <= distance)
        {
            rows.Add(newobject[] { (long)reader.GetValue(0), currentDistance });
        }
    }
    return rows;
}

[SqlFunction(
    FillRowMethodName = "FillRowJoinFilterQuery"

```

```

        , TableDefinition = "oid1 bigint, oid2 bigint"
        , DataAccess = DataAccessKind.Read
        , SystemDataAccess = SystemDataAccessKind.Read
    )]
    public static IEnumerable JoinFilterQuery(
        SqlString table_name_1,
        SqlString column_name_1,
        SqlString table_name_2,
        SqlString column_name_2)
    {
        List<Int64[]> oids = newList<Int64[]>();

        using (SqlConnection connection = Helper.ContextConnection)
        {
            connection.Open();
            using (SqlCommand command = connection.CreateCommand())
            {
                string tableName1 = table_name_1.Value;
                string columnName1 = column_name_1.Value;
                string tableName2 = table_name_2.Value;
                string columnName2 = column_name_2.Value;

                string schemaName1 = "";
                string schemaName2 = "";

                Helper.SplitSchemaAndTableName(command, ref tableName1, ref schemaName1);
                Helper.SplitSchemaAndTableName(command, ref tableName2, ref schemaName2);

                int count1 = CountTableRows(command, tableName1, schemaName1);
                int count2 = CountTableRows(command, tableName2, schemaName2);

                conststring tableAlias1 = "gis_t1";
                conststring tableAlias2 = "gis_t2";

                string swappedTableAlias1;
                string swappedTableAlias2;
                string swappedColumnName1;
                string swappedColumnName2;

                if (count1 > count2)
                {
                    swappedTableAlias1 = tableAlias1;
                    swappedTableAlias2 = tableAlias2;
                    swappedColumnName1 = columnName1;
                    swappedColumnName2 = columnName2;
                }
                else
                {
                    swappedTableAlias1 = tableAlias2;
                    swappedTableAlias2 = tableAlias1;
                    swappedColumnName1 = columnName2;
                    swappedColumnName2 = columnName1;
                }

                string primaryKey1 = GeometryTable.GetPrimaryKeyColumn(command, schemaName1,
                    tableName1);
                string primaryKey2 = GeometryTable.GetPrimaryKeyColumn(command, schemaName2,
                    tableName2);

                Helper.BuildCommand(command, string.Format(
                    @"SELECT [{0}].[{10}], [{1}].[{11}]

```

```

        FROM [{2}].[{3}] as [{0}], [{4}].[{5}] as [{1}]
        WHERE (
            [{6}].[{7}_Envelope_MinX] < ST.MaxX([8].[9]) AND
            [{6}].[{7}_Envelope_MaxX] > ST.MinX([8].[9]) AND
            [{6}].[{7}_Envelope_MinY] < ST.MaxY([8].[9]) AND
            [{6}].[{7}_Envelope_MaxY] > ST.MinY([8].[9])
        )"
        , tableAlias1
        , tableAlias2
        , schemaName1
        , tableName1
        , schemaName2
        , tableName2
        , swappedTableAlias1
        , swappedColumnName1
        , swappedTableAlias2
        , swappedColumnName2
        , primaryKey1
        , primaryKey2
    ));

using (SqlDataReader reader = command.ExecuteReader())
{
    while (reader.Read())
    {
        oids.Add(new Int64[] { (long)reader.GetValue(0),
            (long)reader.GetValue(1) });
    }
}

return oids;
}

private static int CountTableRows(SqlCommand command, string tableName, string
schemaName)
{
    Helper.BuildCommand(command, string.Format(
@"SELECT count(*)
    FROM [{0}].[{1}]"
        , schemaName, tableName));
    return (int)command.ExecuteScalar();
}

public static void FillRowJoinFilterQuery(Object obj, out SqlInt64 oid1, out SqlInt64
oid2)
{
    oid1 = new SqlInt64(((Int64[])obj)[0]);
    oid2 = new SqlInt64(((Int64[])obj)[1]);
}

[SpatialQuery]
[SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
public static IEnumerable RelateQuery(
    SqlString table_name,
    SqlString column_name,

```

```

SqlBytes ewkb,
SqlString intersectionPattern)
{
return SpatialQuery.Execute(table_name, column_name, ewkb,
new SpatialQuery.Arguments(intersectionPattern), RelateQueryHandler);
}

[ SpatialQuery ]
[ SqlFunction(
    FillRowMethodName = SpatialQuery.FillRow
    , TableDefinition = SpatialQuery.TableDefinition
    , DataAccess = DataAccessKind.Read
    , SystemDataAccess = SystemDataAccessKind.Read
)]
public static IEnumerable RelateQueryWhere(
SqlString table_name,
SqlString column_name,
SqlBytes ewkb,
SqlString intersectionPattern,
SqlString condition)
{
return SpatialQuery.Execute(table_name, column_name, ewkb, condition,
new SpatialQuery.Arguments(intersectionPattern), RelateQueryHandler);
}

private delegate bool RelateFunction(IGeometry g);

private static IEnumerable RelateQueryHandler(SqlDataReader reader, IGeometry geometry,
SpatialQuery.Arguments args)
{
string intersectionPattern = args.string1.Value;

RelateFunction function;
switch (intersectionPattern.Trim().ToLower())
{
case "contains":
function = geometry.Contains;
break;
case "coveredby":
function = geometry.CoveredBy;
break;
case "covers":
function = geometry.Covers;
break;
case "crosses":
function = geometry.Crosses;
break;
case "disjoint":
function = geometry.Disjoint;
break;
case "equals":
function = geometry.Equals;
break;
case "equalsexact":
function = geometry.EqualsExact;
break;
case "intersects":
function = geometry.Intersects;
break;
case "overlaps":
function = geometry.Overlaps;

```

```

break;
case "touches":
    function = geometry.Touches;
break;
case "within":
    function = geometry.Within;
break;
default:
    function = null;
break;
    }

List<Int64> oids = newList<Int64>();
if (function == null)
while (reader.Read())
    {
        IGeometry _geometry = ConvertGeometry.ToGeometry(reader.GetSqlBytes(1));
        if (geometry.Relate(_geometry, intersectionPattern))
            {
                oids.Add((long)reader.GetValue(0));
            }
    }
else
while (reader.Read())
    {
        IGeometry _geometry = ConvertGeometry.ToGeometry(reader.GetSqlBytes(1));
        if (function(_geometry))
            {
                oids.Add((long)reader.GetValue(0));
            }
    }
return oids;
    }
}

public class SpatialQueryAttribute : Attribute
{
    public string Columns;
}

```

Ek 2 Uygulamaveritabanı SQL komutları

Veritabanı oluşturma SQL komutları:

```
USE [master]
```

```
GO
```

```
CREATEDATABASE [test] ONPRIMARY
```

```
( NAME =N'test',FILENAME=N'C:\Program Files\Microsoft SQL  
Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\test.mdf', SIZE =1024KB , MAXSIZE  
=UNLIMITED, FILEGROWTH = 1024KB )
```

```
LOGON
```

```
( NAME =N'test_log',FILENAME=N'C:\Program Files\Microsoft SQL  
Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\test_log.ldf', SIZE =1024KB , MAXSIZE  
= 2048GB , FILEGROWTH = 10%)
```

```
GO
```

```
ALTERDATABASE [test] SETCOMPATIBILITY_LEVEL= 100
```

```
GO
```

```
IF (1 =FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
```

```
begin
```

```
EXEC [test].[dbo].[sp_fulltext_database]@action ='enable'
```

```
end
```

```
GO
```

```
ALTERDATABASE [test] SETANSI_NULL_DEFAULTOFF
```

```
GO
```

```
ALTERDATABASE [test] SETANSI_NULLSOFF
```

```
GO
```

```
ALTERDATABASE [test] SETANSI_PADDINGOFF
```

```
GO
```

```
ALTERDATABASE [test] SETANSI_WARNINGSOFF
```

```
GO
```

```
ALTERDATABASE [test] SETARITHABORTOFF
```

```
GO
```

```
ALTERDATABASE [test] SETAUTO_CLOSEOFF
```

```
GO
```

```
ALTERDATABASE [test] SETAUTO_CREATE_STATISTICSON
```

```
GO
```

```
ALTERDATABASE [test] SETAUTO_SHRINKOFF
```

```

GO
ALTERDATABASE [test] SETAUTO_UPDATE_STATISTICSON
GO
ALTERDATABASE [test] SETCURSOR_CLOSE_ON_COMMITOFF
GO
ALTERDATABASE [test] SETCURSOR_DEFAULTGLOBAL
GO
ALTERDATABASE [test] SETCONCAT_NULL_YIELDS_NULLOFF
GO
ALTERDATABASE [test] SETNUMERIC_ROUNDABORTOFF
GO
ALTERDATABASE [test] SETQUOTED_IDENTIFIEROFF
GO
ALTERDATABASE [test] SETRECURSIVE_TRIGGERSOFF
GO
ALTERDATABASE [test] SETDISABLE_BROKER
GO
ALTERDATABASE [test] SETAUTO_UPDATE_STATISTICS_ASYNCOFF
GO
ALTERDATABASE [test] SETDATE_CORRELATION_OPTIMIZATIONOFF
GO
ALTERDATABASE [test] SETTRUSTWORTHYOFF
GO
ALTERDATABASE [test] SETALLOW_SNAPSHOT_ISOLATIONOFF
GO
ALTERDATABASE [test] SETPARAMETERIZATIONSIMPLE
GO
ALTERDATABASE [test] SETREAD_COMMITTED_SNAPSHOTOFF
GO
ALTERDATABASE [test] SETHONOR_BROKER_PRIORITYOFF
GO
ALTERDATABASE [test] SETREAD_WRITE
GO
ALTERDATABASE [test] SETRECOVERYFULL
GO
ALTERDATABASE [test] SETMULTI_USER
GO
ALTERDATABASE [test] SETPAGE_VERIFYCHECKSUM
GO
ALTERDATABASE [test] SETDB_CHAININGOFF

```

```
GO
```

Yönetim tablolarının oluşturulmasında kullanılan SQL komutları:

```
USE [test]
```

```
GO
```

```
SETANSI_NULLSON
```

```
GO
```

```
SETQUOTED_IDENTIFIERON
```

```
GO
```

```
CREATETABLE [dbo].[KYCM] (  
    [Id] [uniqueidentifier] NOTNULL,  
    [TabloAdi] [nvarchar](64)NOTNULL,  
    [AlanAdi] [nvarchar](64)NOTNULL,  
    [YetkiTuru] [tinyint] NOTNULL,  
CONSTRAINT [PK_KYCM] PRIMARYKEYCLUSTERED  
(  
    [Id] ASC  
)WITH  
(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCK  
S=ON,ALLOW_PAGE_LOCKS=ON)ON [PRIMARY]  
)ON [PRIMARY]
```

```
GO
```

```
ALTERTABLE [dbo].[KYCM] ADDCONSTRAINT [DF_KYCM_Id] DEFAULT (newid())FOR  
[Id]
```

```
GO
```

```
ALTERTABLE [dbo].[KYCM] ADDCONSTRAINT [DF_KYCM_YetkiTuru] DEFAULT ((0))FOR  
[YetkiTuru]
```

```
GO
```

Tablo_Geometry test tablosunun oluşturulması için kullanılan SQL komutları:

```
USE [test]
```

```
GO
```

```
SETANSI_NULLSON
```

```
GO
```

```
SETQUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE TABLE [dbo].[Tablo_Geometry] (
    [Id] [uniqueidentifier] NOT NULL,
    [NesneAdi] [nvarchar](50) NULL,
    [Konum] [geometry] NULL,
CONSTRAINT [PK_Tablo_Geometry] PRIMARY KEY CLUSTERED
(
    [Id] ASC
) WITH
(PAD_INDEX=OFF, STATISTICS_NORECOMPUTE=OFF, IGNORE_DUP_KEY=OFF, ALLOW_ROW_LOCKS=ON, ALLOW_PAGE_LOCKS=ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
```

```
ALTER TABLE [dbo].[Tablo_Geometry] ADD CONSTRAINT [DF_Table_1_id] DEFAULT
(newid()) FOR [Id]
```

```
GO
```

Tablo_Geography test tablosunun oluşturulması için kullanılan SQL komutları:

```
USE [test]
```

```
GO
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE TABLE [dbo].[Tablo_Geography] (
    [Id] [uniqueidentifier] NOT NULL,
    [NesneAdi] [nvarchar](50) NULL,
    [Konum] [geography] NULL,
CONSTRAINT [PK_Tablo_Geography] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
```

```

) WITH
(PAD_INDEX=OFF, STATISTICS_NORECOMPUTE=OFF, IGNORE_DUP_KEY=OFF, ALLOW_ROW_LOCK
S=ON, ALLOW_PAGE_LOCKS=ON) ON [PRIMARY]
) ON [PRIMARY]

```

GO

```

ALTERTABLE [dbo].[Tablo_Geography] ADDCONSTRAINT [DF_Table_2_id] DEFAULT
(newid()) FOR [Id]

```

GO

YB_Tablolari_Olustur SQL komutları:

```

SETANSI_NULLSON

```

GO

```

SETQUOTED_IDENTIFIERON

```

GO

```

CREATEPROCEDURE [dbo].[YB_Tablolari_Olustur]

```

```

    @TabloAdi asnvarchar(64),

```

```

    @AlanAdi asnvarchar(64),

```

```

    @YetkiTuru astinyint

```

AS

```

BEGIN

```

```

    SETNOCOUNTON;

```

```

    Declare @TableVarmi asint;

```

```

    Select @TableVarmi =COUNT(*) FromsysobjectsWhere xtype='u'and name =

```

```

@TabloAdi;

```

```

    if @TableVarmi = 0

```

```

        Returnnull;

```

```

    Declare @AlanVarmi asint;

```

```

    Select @AlanVarmi =COUNT(*) FromINFORMATION_SCHEMA.COLUMNSWhere

```

```

TABLE_NAME = @TabloAdi and COLUMN_NAME = @AlanAdi;

```

```

    If @AlanVarmi = 0

```

```

        Return 0;

```

```

    Declare @KYBTable asnvarchar(1000);

```

```

    Set @KYBTable ='CREATE TABLE [dbo].[KYB_'+ @TabloAdi +'] (

```

```

        [Id] [uniqueidentifier] NOT NULL,

```

```

        [Adi] [nvarchar](64) NOT NULL,

```

```

        [Turu] [tinyint] NOT NULL,

```

```

        [KYBGeometry] [geometry],

```

```

        [KYBGeography] [geography]
        CONSTRAINT [PK_KYB_'+ @TabloAdi +'] PRIMARY KEY CLUSTERED
        ( [Id] ASC ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
        [PRIMARY]) ON [PRIMARY]';

```

```

Declare @KYBKITable as nvarchar(1000);
Set @KYBKITable = 'CREATE TABLE [dbo].[KYBKI_'+ @TabloAdi +'] (
        [Id] [uniqueidentifier] NOT NULL,
        [KYBId] [uniqueidentifier] NOT NULL,
        [KullaniciAdi] [nvarchar](64) NOT NULL,
        [KYEkle] [bit],
        [KYDuzenle] [bit],
        [KYSil] [bit]
        CONSTRAINT [PK_KYBKI_'+ @TabloAdi +'] PRIMARY KEY CLUSTERED
        ( [Id] ASC ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
        [PRIMARY]) ON [PRIMARY]
';
Exec(@KYBTable);
Exec(@KYBKITable);
Exec('ALTER TABLE [dbo].[KYB_'+ @TabloAdi +'] ADD CONSTRAINT
[DF_KYB_'+ @TabloAdi + '_Id] DEFAULT (newid()) FOR [Id]');
Exec('ALTER TABLE [dbo].[KYBKI_'+ @TabloAdi +'] ADD CONSTRAINT
[DF_KYBKI_'+ @TabloAdi + '_Id] DEFAULT (newid()) FOR [Id]');

Exec('YB_TV_F_OLUSTUR' + @TabloAdi + ', ' + @AlanAdi + ', '
+ @YetkiTuru);
Return 1;

```

END

GO

YB_Ekle SQL komutları:

SETANSI_NULLSON

GO

SETQUOTED_IDENTIFIERON

GO

```

CREATEPROCEDURE [dbo].[YB_Ekle]
        @TabloAdi nvarchar(64),
        @YetkiBolgesiAdi nvarchar(64),

```

```

        @Turu tinyint,
        @KYBGeometry geometry,
        @KYBGeography geography
AS
BEGIN
    Declare @MyId uniqueidentifier;
    Set @MyId =newid();
    Declare @strInsert asnvarchar(1000);
    Set @strInsert ='Insert Into KYB_'+ @TabloAdi +'
(Id,Adi,Turu,KYBGeometry,KYBGeography)Values (''+Cast(@MyId
asvarchar(50))+'',''+ @YetkiBolgesiAdi +''',''+Cast(@Turu
aschar(1))+'',geometry::STGeomFromText('+ @KYBGeometry.ToString()+',0),
'+',geography::STGeomFromText('+ @KYBGeography.ToString()+',0)) '

    EXEC (@strInsert);
    Select @MyId as Id
END
GO

```

ÖZGEÇMİŞ

Doğum tarihi 03.10.1978

Doğum yeri Diyarbakır

Lise 1989-1996 Şanlıurfa Anadolu Lisesi

Lisans 1996-2000 Yıldız Teknik Üniversitesi İnşaat Fak.
Jeodezi ve Fotogrametri Mühendisliği Bölümü

Yüksek Lisans 2000-2003 Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü
Jeodezi ve Fotogrametri Müh. Anabilim Dalı

Doktora 2003- Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü
Jeodezi ve Fotogrametri Müh. Anabilim Dalı

Çalıştığı kurum(lar)

2001-2006 YTÜ Fen Bilimleri Enstitüsü Araştırma Görevlisi