MEKANSAL VERİTABANLARINDA HIZLI SORGULAMA

ARZU KÜTÜKCÜ

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Bölümü

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ARALIK 2009

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç.Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Arzu KÜTÜKCÜ tarafından hazırlanan MEKANSAL VERİTABANLARINDA HIZLI SORGULAMA adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Osman ABUL

Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Erdoğan DOĞDU _____

Üye : Yrd. Doç. Dr. Bülent GÜMÜŞ _____

Üye : Yrd. Doç. Dr. Osman ABUL _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Arzu KÜTÜKCÜ

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri Enstitüsü**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Yrd. Doç. Dr. Osman ABUL**
Tez Türü ve Tarihi : **Yüksek Lisans – Aralık 2009**

Arzu KÜTÜKCÜ

MEKÂNSAL VERİTABANLARINDA HIZLI SORGULAMA

ÖZET

Mekânsal verilerin kullanımı gün geçtikçe artmakta, kullanıldığı alanlar da çeşitlilik kazanmaktadır. Bu verilere hızlı ulaşabilmenin yolları incelenmeye devam etmekte, veritabanları üzerinde kullanılan mekansal fonksiyonlar ve indekslerin kullanımı da önem kazanmaktadır.

Bu çalışmanın amacı, seçilen bir veritabanı üzerinde kullanılan geometri tipleri, hızlı erişim için desteklenen fonksiyonlar, indeksler ve operatörleri incelemek, Spatial(mekânsal) indeks kullanımı, fonksiyon ve operatörlerin analizlerdeki önemi, veriye daha hızlı erişim için sorguların tasarımında dikkat edilmesi gereken noktalar belirlenmeye çalışıltırılmıştır. Tez kapsamında ASP.NET teknolojisi kullanılarak geliştirilen uygulama kullanılarak Akım Gözlem İstasyonu, Akarsu, Nehir, Göl ve Baraj gibi mekansal veriler için, yaygın olarak kullanılan sorgular tasarlanmıştır. Geliştirilen uygulamada Google Maps ve Geoserver gibi web servisleri üzerinden sunulan haritalar da altlık olarak kullanılmıştır.

Mekânsal veriler üzerinde kullanımına izin verilen indekslerin performansı, çalıştırıldıklarında kullandıkları işlemci zamanı, üretilen sorgu planları kullanılarak değerlendirilmiştir. Yapılan testler sonucunda operatör seçimi ve mekansal indeks parametreleri kullanımının veriye hızlı erişim için hayati öne taşıdığı görülmüştür.

Anahtar Kelimeler: Mekânsal Veri, Mekânsal Analiz, Oracle Spatial, ASP.NET, Web Servisleri

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Asst. Prof. Dr. Osman ABUL**
Degree Awarded and Date : **M.Sc. – December 2009**

Arzu KÜTÜKCÜ

EFFICIENT QUERYING OF SPATIAL DATABASES

ABSTRACT

The spatial data is becoming more widely used from day to day and its area of usage also gains variety. The examinations continue to find easy ways for accessing this kind of data and it gains more importance to use spatial functions and indexes which are used on databases.

The aim of this study is to examine the geometry types used on a specific database; functions, indexes and operators supported for rapid access; spatial index usage; the importance of functions and operators for analyses and to determine the important points on the design of queries to create more rapid access to data. Within the scope of this thesis, queries widely used for spatial data such as Stream Gauging Station, River, Stream, Lake and Dam will be designed through the application improved via ASP.NET technology. Geoserver and Google Maps services are used as a resource for underlay display on the web sites accessed via this application.

The performance of the indexes allowed to be used on spatial data by the database will be evaluated according to CPU time and the explain plans produced when they are operated on SQL statements. As a result, these efforts indicate that the use of proper spatial operations and database indexing are vital for fast accessing.

Keywords: Spatial Data, Spatial analysis, Oracle Spatial, ASP.NET, Web Services

TEŞEKKÜR

Kıymetli tecrübelerini benimle paylaşan tez hocam Yrd.Doç.Dr. Osman ABUL'a, TOBB ETU Bilgisayar Mühendisliği Bölümü'nün değerli Öğretim Üyelerine teşekkür ederim.

Çalışmamda kullandığım gerçek ve güncel verileri bana sağlayan, mensubu olmaktan onur duyduğum Devlet Su İşleri Genel Müdürlüğü'ne ve DSİ Coğrafi Bilgi Sistemleri Şube Müdürü Kemal SEYREK'e teşekkür ederim.

Bu çalışma süresince beni destekleyen aileme, özellikle geçirdiği ciddi sağlık sorununa rağmen manevi desteğini üzerimden eksik etmeyen sevgili dayım İlhami ÇETİN'e, Devlet Su İşleri Genel Müdürlüğü çalışanı sevgili arkadaşım Emine GÖKYAPRAK'a, TOBB ETU asistanlarından değerli arkadaşım Elif Tuğçe ÖRS'e çalışmam boyunca bana verdikleri destek için teşekkür ederim.

İÇİNDEKİLER

ÇİZELGE LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
OGC	Open Geospatial Consortium
W3C	World Wide Web Consortium
SDO	Spatial Data Option
RDBMS	Relational Database Management System
AGİ	Akım Gözlem İstasyonu
SDO	Spatial Data Option
OEM	Oracle Enterprise Manager
CBO	Cost Based Optimizer
SQL	Structured Query Language
WFS	Web Feature Service
WMS	Web Map Service
WCS	Web Coverage Service
DOM	Document Object Model
SAX	Simple API for XML
XML	eXtensible Markup Language
KML	Keyhole Markup Language
GML	Geography Markup Language
HTML	Hyper Text Markup Language
XML	Extensible Markup Language
DTD	Document Type Definition
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UDDI	Universal Description, Discovery, and Integration
WSDL	Web Service Definition Language
SOAP	Simple Object Access Protocol
DCOM	Distributed Component Object Model
CORBA	Common Object Request Broker Architecture
RMI	Remote Method Invocation
DEM	Digital Elevation Model
NASA	National Aeronautics and Space Administration
SRTM	Shuttle Radar Topography Mission

BÖLÜM 1

GİRİŞ

Günümüzde, bilgi teknolojilerinin kullanımı hayatımızın her alanında yaşantımıza girmiştir. Özellikle son yıllarda, gerek navigasyon uygulamalarının yaygınlaşması, Virtual Earth, Google Maps gibi uygulamalar vasıtasıyla bilmediğimiz bir adrese yönelik sorgulamaların internet üzerinden yapılarak gerek adres, gerekse de ilgili diğer verilere rahatlıkla internet üzerinden ulaşılabilmesi coğrafi verilerin günlük hayatta kullanılmasını hızlandırmıştır. Bu gelişmeler beraberinde mekansal verilerin üretilmesi, güncellenmesi, depolanması ve verinin paylaşılması sorunlarını beraberinde getirmiştir.

Coğrafi veriler ile bu verilere ait öznitelik verilerinin üretilmesi, güncellenmesi ve ilişkisel bir veritabanı yapısında depolanarak analiz edilmesi çalışmaları ilk başlarda Coğrafi Bilgi Sistemi (CBS) adı altında ayrı bir çalışma konusu olarak gelişmesini sürdürmüştür. Diğer taraftan büyük hacimli verilerin mantıksal bir yapıda depolanmasını, bir bilgisayar ağı yapısı içerisinde kullanılmasını sağlayacak İlişkisel Veritabanı Yönetim Sistemleri (RDBMS) ayrı çalışma konuları olarak gelişmiştir. Ancak son yıllardaki iletişim, internet teknolojileri ile kullanıcı ihtiyacı konularındaki gelişmelerle beraber coğrafi veriler ile bu coğrafi verilere ait diğer tüm öznitelik verilerinin bütünleşik bir yapıda tek bir İlişkisel Veritabanı Yönetim Sistemi (RDBMS) içinde birlikte tutulacağı ve yönetiminin sağlanacağı çözümler geliştirilmeye başlanmıştır. Halihazırda, ESRI tarafından "Coğrafi Veritabanı Sunucusu" olarak geliştirilen ArcSDE programının yanında gerek ticari gerekse de açık kaynak kodlu olarak Oracle, Microsoft SQL Server, DB2, PostgreSQL, mySQL gibi değişik İlişkisel Veritabanı Yönetim Sistemlerine (RDBMS) ait "Spatial" modulleri geliştirilmiştir. Söz konusu "İlişkisel Veritabanı Yönetim Sistemleri" bu konuda değişik çözümler sunarak "Cografi Veritabanı Sunucusu" olarak birçok çalışmada kullanılmaya başlanmıştır.

Coğrafi verilerin istemci/sunucu yapısında saklanması ve bilgisayar ağı üzerinden çok sayıda kullanıcı tarafından kullanılmaya başlanması önemli bir gelişme olmakla

birlikte beraberinde veriye hızlı ulaşım, güvenlik ve performans problemlerini getirmiştir. Diğer taraftan "İlişkisel Veritabanı Yönetim Sistemlerinin" geliştirilmesine paralel olarak, normal veritabanı yeteneklerinin coğrafi veriler üzerinde de uygulanabilmesi ve yeni özelliklerin eklenmesi çalışmaları devam etmektedir.

İlişkisel Veritabanı Yönetim Sistemlerinde (RDBMS) performans artışının sağlanması amacıyla çokboyutlu veriye erişim ve mekansal veriyi indeksleme metodları, yapılan çalışmalarla geliştirilmekte olup bu metodlara "R-trees", "R^+ trees", "K-D-B-trees", "2D Isam" örnek olarak verilebilir[9]. İndeksleme yöntemleri ve mekansal veriye hızlı erişim çalışmaları devam eden ve sürekli gelişen bir alandır. Bu çalışma kapsamında, yapılan araştırmalar sonucu söz edilen metodlar arasında en başarılı bulunan R-tree ve R-tree indeksinin bir varyantı olan R^*-tree indeks incelenecektir.

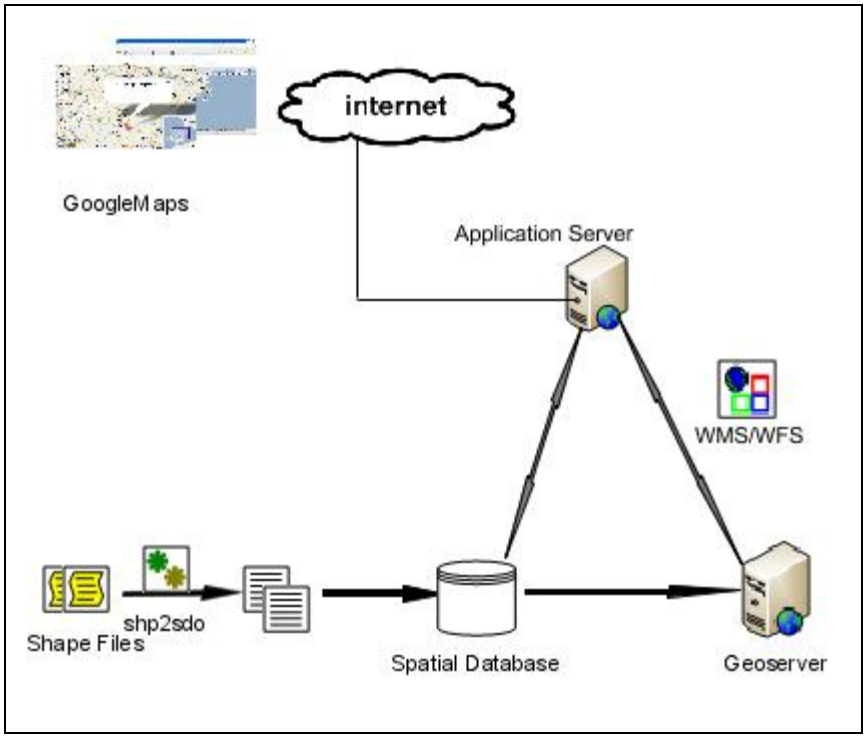
Bu çalışmada, coğrafi verilerin kullanılması sırasında karşılaşılacak problemlerin gözlenmesi ve test edilmesi seçilen Oracle RDBMS ile "Spatial" modülü kullanılarak yapılacak olup, söz konusu veritabanı yönetim sisteminin coğrafi veriler için getirdiği özelliklerden yararlanarak bu tür verilere daha hızlı erişim yolları aranmaya çalışılmıştır. Ayrıca, çalışma kapsamında saklanan coğrafi veriler kullanılarak; "Belirlenen noktaya en yakın komşularının bulunması", "seçilen noktaya verilen mesafe içinde kalan geometrilerin bulunması" gibi analizler ve sorgulamalar yapılarak bu sorgulamaların sonucunda CPU ve I/O değerleri elde edilip, performans sonuçları test edilecek ve performansın söz konusu değerler doğrultusunda iyileştirilmesi çalışmaları yapılacaktır.

1.1. Çalışmanın Tanımı

Bu tez kapsamında;

1. Hazırlanan coğrafi veri katmanları (havza, akarsu, AGİ, göl vb…) kullanılacak ara yazılımlar ile Oracle veritabanına aktarılacak ve kullanıcı tarafından herhangi bir program yüklenilmesine gerek kalmadan internet tarayıcısı üzerinden çalışacak bir uygulama yazılımı hazırlanacaktır.

2. Hazırlanan uygulama yazılımı ile, veritabanına aktarılan coğrafi veriler kullanılarak değişik coğrafi sorguların geliştirilmesine ve bu sorgular üzerinde çeşitli performans değerlendirilmesi yapılarak iyileştirme ve optimizasyon çalışmalarının yapılması amaçlanmıştır. Çalışma kapsamında kullanılan sistemin genel yapısı Şekil 1.1.'de verilmektedir.



Şekil 1.1. Sistemin Genel Yapısı

Bu yapıya göre sistemi genel olarak incelediğimizde;

1. Verilerin veritabanına atılması sırasında öncelikle Shapefile formatında elde edilen veriler "sh2sdo.exe" programı kullanılarak "Oracle Spatial"ın özel "SDO_GEOMETRY" veri tipine dönüştürülmüş ve dönüşümü yapılan veriler "Sql loader" programı kullanılarak Oracle veritabanına aktarılmıştır.

2. Uygulama kapsamında "Mekânsal Sunucu" olarak java tabanlı açık kaynak kodlu "Geoserver" programı kullanılmıştır. Söz konusu program, "Mekânsal Veritabanı Sunucusu" olarak kullanılan "Oracle Spatial"dan verileri elde etmekte ve alınan verilerin "OpenGIS Consorsium-OGC" tarafından standartları tespit edilmiş olan WMS, WFS, WCS ve benzeri "Web Servisleri"ne dönüştürülmesinde

kullanılmaktadır. Bu çalışmada, "Oracle Spatial"da tutulan Havza geometrileri "Geoserver" üzerinden WMS Servisine çevrilmekte ve geliştirilen ASP.NET uygulamasında söz konusu servis altlık olarak kullanılmaktadır.

3. Son kullanıcı tarafından kullanılacak sunum katmanı ise Microsoft .Net Framework, C#, ASP.NET, Google API ve OpenLayers API gibi uygulama geliştirme araçları kullanılarak hazırlanmıştır. Geliştirilen sunum katmanında, daha önce "Oracle Spatial" ve "GeoServer" kullanılarak "XML", "KML" GML" formatlarında hazırlanan coğrafi veriler ile diğer web servisleri "OpenLayers API" kullanılarak birleştirilmiş ve hazırlanılan "ASP.NET" web sayfasında yayımlanmıştır.

4. Oracle'dan "GML" formatında alınabilen veriler geliştirilen Oracle prosedür ve fonksiyonları yardımıyla istenen formata dönüştürülmüştür.

5. ASP.NET teknolojisi kullanılarak geliştirilen uygulama üzerinden yapılan sorgulamalar ile Türkiye üzerinde belirlenmiş 26 havza için "Havza içerisinde Kalan Akım gözlem istasyonlarının bulunması", "belirlenen bir Akım Gözlem İstasyonuna-AGİ en yakın istasyonların saptanması", "Nehirlere Yakın Olan AGİ"lerin saptanması gibi sorgulamalar çalıştırılmıştır. Ayrıca, Oracle tarafından sağlanan "Network Data Model"in sağlamış olduğu ve coğrafi verinin yönlendirilmiş çizge (directed graph) şeklinde ifade edilmesi yöntemi ile seçilen bir nehir üzerinde (Bu tez çalışmasında Fırat nehri seçilmiştir) belirlenen bir noktanın yukarısında ve aşağısında kalan AGİ'lerin tespiti çalışması yapılmıştır.

Bu çalışmalar sonucunda, hem veritabanı tarafında hem de uygulama tarafında yapılması gereken işlemler anlatılarak coğrafi verilerle geliştirilecek projeler için bir kaynak oluşturulmaya çalışılmış, mekansal verilere erişilirken önem arzeden hızın artırılması için çeşitli yöntemler geliştirilmiştir.

BÖLÜM 2

COĞRAFİ VERİLERİN DEPOLANMASI VE SUNUMUNDA KULLANILAN TEKNOLOJİ ALTYAPISI

Bu bölümde Coğrafi verilerin sunulmasında kullanılan mimari altyapılar, halihazırda kullanılan web servisleri ve veri standartları anlatılmış olup, bu tez çalışmasında kullanılan oracle spatial veritabanı, veri formatı ve web servislerinin seçiminde rol oynayan faktörler açıklanmıştır.

2.1. Coğrafi Veri Sunucu Katmanında Kullanılan Mimari Altyapı

Başlangıçta, üretilen mekansal veriler ile bu verilere ait öznitelik verileri kişisel bilgisayarlarda dosya tabanlı olarak saklanmaktayken süreç içerisinde bu verilerin depolanması için veritabanlarının kullanımı ortaya çıkmış ve ilk etapta MS Access gibi kişisel veritabanları kullanılmaya başlanılmıştır. Daha çok kişisel uygulamalarda söz konusu saklama sistemleri yeterli olurken, çok kullanıcılı ortamda eldeki mevcut coğrafi verilerin kullanılması amacıyla İlişkisel Veritabanı Yönetim Sistemlerinin kullanılması zorunluluğu doğmuştur. Başlarda mekansal verilere ait grafik veriler dosya tabanlı olarak, bu verilere ait öznitelik verileri ise İlişkisel Veritabanı Yönetim Sistemleri kullanılarak saklanmış ve bir UniqueID kullanılarak bu veriler arasında ilişkisel bağlantılar kurulmuş ve eldeki verilerin bu yöntem ile yönetilmesi bir çözüm olarak sunulmuştur. Halen bazı yazılımlar tarafından söz konusu çözüm, coğrafi verilerin üretilmesinde ve yönetilmesinde kullanılmakla birlikte performans ve kullanım açısından istenilen yeterliliğe sahip olunamamıştır. Dolayısıyla, hem grafik hem de öznitelik verilerinin ilişkisel olarak tek bir veritabanında depolanması ve paylaşılmasına yönelik çalışmalara başlanılmış ve ilk etapta ESRI firması tarafından ArcSDE geliştirilmiştir. Tüm ilişkisel veritabanı yönetim sistemlerini destekleyen bu yazılımla beraber zaman içerisinde hemen hemen tüm İlişkisel Veritabanı Yönetim Sistemi geliştiren firmalar tarafından, coğrafi verileri de normal veritabanı uygulamaları ile birlikte aynı ortamda saklayacak ve yönetecek çözümler kullanıcılara sunulmaya başlamıştır.

Halihazırda, hem ticari firmalar tarafından üretilen İlişkisel Veritabanı Sistemleri hemde açık kaynak kodlu olarak geliştirilen İlişkisel Veritabanı yönetim Sistemleri coğrafi verileri de diğer verilerle birlikte kullanacak ve yönetecek araçlara sahip bulunmaktadır. Aşağıda söz konusu çözümler kısaca verilmektedir.

ArcSDE

ArcSDE yazılımı RDBMS içerisinde saklanan coğrafi verileri sorgulamak, analiz etmek ve düzenlemek amacıyla kullanılan ESRI ArcGIS ailesi ürünlerindendir. IBM DB2, Informix, Microsoft SQL Server ve Oracle üzerindeki coğrafi verilerin ArcView, ArcEditor ve ArcInfo gibi ArcGIS Desktop ürünlerine bilgi sunulmasını sağlayan önemli bir bileşendir. Coğrafi veri için ortak bir model sunmaktadır. Mekansal veriler, RDBMS için önem arzeden veri import ve export programları kullanmak suretiyle, bir RDBMS'ten başka birine veri kaybı olmadan kolaylıkla taşınabilmektedir. ArcSDE, nokta (point), çizgi (line) ve alan (polygon) gibi geometrik veriler veritabanına eklenirken bozuk yapılı geometrilerin eklenmesine izin vermeyerek veri bütünüğünü de korumaktadır.

Oracle Spatial

Oracle Spatial, Spatial olarak da adlandırılan mekansal verilerin sorgulanması, güncellenmesi ve depolanmasına yardımcı olan SQL şema ve fonksiyonları içeren teknolojidir. OGC standartlarını desteklemektedir.

Spatial teknoloji;

1. MDSYS Şeması,
2. Spatial indeks mekanizması,
3. Spatial analiz işlemleri için fonksiyonlar, prosedürler ve operatörler,
4. Tuning işlemleri için operatörler,
5. Topoloji veri modeli,
6. Network data model oluşturulması,
7. GeoRaster veriler üzerinde analizlerin yapılabilmesi,

gibi özellikleri içerir.

Tanımlanan spatial indeks ile 2, 3 veya 4 boyutlu veri indekslenebilmekte, gelişmiş indeksleme özellikleri ve operatörlerle etkin bir şekilde mekansal veriye erişilebilmektedir [2].

Microsoft SQL Server 2008 Spatial

Microsoft SQL Server 2008 ile gelen mekansal verinin desteklenmesi özelliği ile, verilerin saklanması için veri tipleri sunulmakta, bu tür verilerin sorgulanması ve analizi için çeşitli olanaklar sağlanmaktadır. Sunulan coğrafi veri kabiliyetleri yardımı ile coğrafi sorgular kolay ve hızlı bir şekilde yapılabilmektedir.

IBM DB2

Mekansal verinin saklanması, sorgulanması ve analizi için IBM firması tarafından IBM DB2 Spatial Extender sunulmuş olup, coğrafi veriler nokta (point), çizgi (line) ve alan (poligon) gibi geometriler şeklinde tanımlanabilir. Coğrafi verinin alınması, veritabanına eklenmesi ve indekslenmesi gibi işlemler için SQL erişimine izin verilir. ESRI ArcExplorer gibi görüntüleme araçlarının kullanımı mümkün olmaktadır. OGC tarafından belirlenen standart ve şartları desteklemektedir.

PostGreSQL/PostGis

"PostgreSQL, POSTGRES'in zengin veri tiplerini ve güçlü veri modelini tutarken, SQL'in geliştirilmiş alt kümesi olan PostQuel dilini kullanır. PostgreSQL ücretsizdir ve kaynak kodu açık dağıtılır" [3] . Açık kaynak kodlu spatial teknoloji için geliştirilen PostGis exlentisi ise PostgreSQL için coğrafi verilerin saklanması, koordinat dönüşümleri ve temel topoloji işlemlerinin yapılmasına izin verir. OGC tarafından belirlenen standart ve şartları desteklemektedir.

MySQL

OGC tarafından belirlenen standart ve şartları destekleyen, coğrafi verilerin saklanması, indekslenmesi ve analizine imkân veren açık kaynak kodlu bir veritabanı yönetim sistemidir.

Yukarıda Coğrafi verilerin "İlişkisel veritabanı Yönetim Sistemleri-RDBMS" içerisinde diğer verilerle birlikte depolanması ve yönetimini sağlayan çözümler

anlatılmış olup, bu çözümlerin avantaj ve dezavantajları Çizelge 2.1.'de verilmektedir.

Çizelge 2.1. Coğrafi verilerin Yönetiminde Yaygın olarak Kullanılan İlişkisel Veritabanı Yönetim Sistemlerinin Karşılaştırması

Spatial Veritabanı	**Avantajları**	**Dezavantajları**
ArcSDE	• Lider CBS sağlayıcı tarafından oluşturulmuş (ESRI) • En iyi CBS işlevselliği	• Fiyatları (tüm sistem için fiyatı 60.000 ABD $'na kadar çıkabilmektedir.)
Oracle Spatial	• Şirketlerin ihtiyaçları için yeterli • Tam olarak aradığımız şey • Önde gelen veritabanı sağlayıcısı • Yeterli destek alınabilir	• Fiyatlar (tüm sistem için fiyatı 50.000 ABD $'na kadar çıkabilmektedir.)
Microsoft SQL Server 2008 Spatial	• Vector veriler için gelişmiş özellikler • Kolay Anlaşılır • Yeterli destek alınabilir	• Geocoding • Network Data Model • Topology Network
IBM DB2 Spatial Extender	• Şirketlerin ihtiyaçları için yeterli • Tam olarak aradığımız şey • Önde gelen veritabanı sağlayıcısı • Yeterli destek alınabilir • ArcExplorer'la çalışır • Fiyatları (free with $30,000 purchase of IBM DB2) ($20,000 < ArcSDE sistem $10,000 < Oracle sistem)	• ArcSDE ile karşılaştırıldığında kısıtlı CBS yetenekleri
PostGIS	• Ücretsiz • Kolaylıkla Download Edilebilir	• Ödenilen Kadar alınır • Son derece küçük bir pazar • Az bilgi sahibi kişiler • Sınırlı Mali Destek • Uzun ömürlü olması risk altında
MySQL	• Açık Kaynak Kodlu • Kolaylıkla Download Edilebilir	• Diğer ticari yazılımlarla karşılaştırıldığında kısıtlı yeteneklere sahip

Yukarıda anlatılan özellikler değerlendirildiğinde bu çalışma kapsamında, Oracle Spatial örnek ugulamamızda kullanılmak üzere seçilmiştir. Elimizdeki mevcut

coğrafi veriler Oracle Spatial kullanılarak Oracle RDBMS'e aktarılacak, hazırlanacak saklı yordamlarla (Stored Procedure) ve fonksiyonlarla ihtiyaç duyulan analizler yapılacaktır.

2.2. CBS Sunucu Katmanında Kullanılan Teknik Altyapı

Gerek eldeki verilerin "İlişkisel Veritabanı Yönetim Sisteminde" saklanılması için standartları sağlamak, gerekse de bu verilerin belli bir standart çerçevesinde sunumunu gerçekleştirmek amacıyla servislerin hazırlanmasına yönelik standartların oluşturulması amacıyla "OpenGIS Consorsium-OGC" kurulmuştur. Halihazırda kullanılan yazılımların tamamı OGC standartlarını karşılamakta ve hazırlanan ürünlerde OGC standartları ile veri yapılarına dikkat edilmektedir. Bu bağlamda "OpenGIS Consorsium" ile mevcut OGC standartları ve World Wide Web Consortium-W3C aşağıda detaylı olarak verilmektedir.

2.2.1. World Wide Web Consortium-W3C

World Wide Web Consortium web için standartların belirlenmesi ve web'in uzun vadeli gelişiminin sağlanması amacıyla oluşturulmuş bir organizasyondur. Başlıca amacı hardware, software, network altyapısı ne olursa olsun bilginin açık bir şekilde paylaşımı için standartların sağlanmasıdır. Bu doğrultuda XML, HTML gibi birçok standardın belirlenmesini sağlamışlardır. Çalışmada da kullanılan bazı w3c standartları aşağıda açıklanmıştır.

Hyper Text Markup Language-HTML'in Yapısı

HTML, web sayfalarını tanımlamak için kullanılan bir dildir. HTML bir programlama dili değil, bir markup (betik) dilidir. <html> şeklinde HTML işaretleme etiketleri (tag) ile tanımlanır ve oluşturulan doküman da web sayfası olarak adlandırılır.

Genişletilebilir işaretleme dili -XML (Extensible Markup Language) Yapısı

XML Standard Generalized Markup Language-SGML'den türemiş, veri iletimi için çok esnek, kullanıcının kolayca anlayabileceği, World Wide Web Consortium-W3C tarafından geliştirilmiş bir dildir[47]. XML, yine XML kullanılarak kendi markup (betik) dilinin tanımlanabilmesini sağlaması sebebi ile bir metamarkup dilidir. XML'in en önemli çıkış noktası, uygulamaların farklı veri formatları kullanıyor olmaları ve XML ile istenen formatta veri üretilebilmesi, bu üretim işleminin de

ortak bir dil kullanılarak yapılabiliyor olmasının veri paylaşımında kolaylıklar sağlamasıdır. XML dokümanını mantıksal ve fiziksel yapı olmak üzere 2 kısıma ayırıyoruz. Mantıksal kısım da Prolog ve Data instance(root eleman ve onun içerdiği elemanlar) olmak üzere kendi içinde 2 kısma ayrılmaktadır.

Prolog

Prolog kısmı XML dokümanının başlangıç kısmıdır. Bir XML dokümanının ana mantıksal kısmıdır. Parser'lar için de ön bir bilgi sağlamaktadır. Prolog Kısmı; XML decleration, Processing Instructions, Document Type Decleration-DTD, yorum(Comment) ve Beyaz Boşluk(White Space) kısımlarından oluşmaktadır. Şekil 2.1.'de görünen ilk dört satır prolog kısmını oluşturmaktadır.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/css" href=" argument.css"?>
<!DOCTYPE arg-conf SYSTEM " http://www.w3.org/TR /arg-conf.dtd">
<!--  A simple XML file -->
<argument-conf>
      <arguments>
            <argument>
    <revision>April-30-2003</revision>
    <author> tobb etu ceng</author>
    <contact>ceng@etu.edu.tr</contact>
            </argument>
            <argument>
    <revision>Sep-16-2003</revision>
    <author>tobb etu eeng</author>
    <contact>eeng@etu.edu.tr</contact>
            </argument>
      </arguments>
</argument-conf>
```

Şekil 2.1. Bir XML dokümanı örneği

XML Deklarasyonu

XML 1.0 dökümanına göre her XML dökümanı bir declarasyon satırı ile başlar. Ve bu iyi biçimlendirilmiş bir XML dökümanında, ilk satırda olmalıdır. XML deklarasyonu aynı zamanda processing instruction'dır. <?xml ... ?> tagları içinde yazılmalıdır. XML deklarasyonu direkt olarak uygulamaya, bunun bir xml dokümanı olduğunu söylemektedir.

İşleme Komutları (Processing Instructions - PI)

İşleme komutları parser'ı geçip direkt uygulama ile iletişimdedirler. "Piname" uygulamaya PI'ın ne tür bir PI olduğunu belirtmektedirler. Pi satırı <?piname pseudo-attributes?> şeklinde bir söz dizimine sahiptir. İşleme komutları küçük harfle yazılmalıdır. Tip kısmında da uygulamanın ne tip bir stil şablonunda (style sheet) işlem yapacağı belirtilmiş olur.

Doküman Tipi Deklarasyonu (Document Type Declaration – DTD)

Şekil 2.7.'de 3. satırda belirtilen Doküman Tipi Deklarasyonu (DTD) satırı parser'a dokümanın ne tür bir doküman olduğunu, veri örneği (data instance) kısmında bulunan elemanların birbirleri ile ilişkilerini belirtir. Doküman Tipi Deklarasyon satırları bir XML dökümanı içerisinde yer alan elemanların tanımlandığı Doküman Tipi Tanımlamalarının (Document Type Definition) parser'a bildirilmesi için kullanılır. DTD'ler dâhili ya da harici olarak tanımlanabilir.

Yorumlar (*Comments*)

Yorumlar uygulama ya da parser'ın dikkate almadığı, ancak dökümanı inceleyen için açıklayıcı bilgilerin bulunduğu kısımlardır. Yorumlar <!-- … --> etiketleri arasına yazılır. Ancak yazılırken metin içerisinde "—" karakteri kullanılmamalıdır. Parser'ın bu karakterleri açıklamanın kapatılması olarak algılaması hataya neden olabilir.

Veri Örneği (*Data Instance*)

XML dokümanında Prolog kısmından sonra Data Instance kısmı başlar. Bir veya daha fazla elemandan oluşur. Elemanlar bir XML dökümanında verilerin tutulduğu temel taşlardır. Veriler etiketler içerisinde yer alır. 3 tip etiket vardır. Başlama, bitiş ve hibrid etiketleridir. Her eleman aynı zamanda eleman'ın ismini de belirten tip kısmı ile başlar ve biter. Her eleman başlama etiketi ile başlar ve bitiş etiketi ile biter. Etiketler "< "işareti ile başlar ve ">" işareti ile de kapanırlar. Bitiş etiketinde "/" işareti de yer alır. Hibrid etiketi ise bu başlama ve bitiş etiketlerinin birleşimi ile oluşmaktadır. Etiketler içerisinde elemanın tipini de ifade eden elemanın ismi yer

alır. Elemanlar isimlendirilirken hataya neden olmamak için bazı kurallara uyulması gerekir. Bunlar;

- Elemanlar sayı ya da noktalama işaretleri ile başlayamazlar.
- Eleman isimleri XML için özel anlamlar içeren "&", "@", "<" gibi sembolleri içermemelidir.
- Eleman isimleri boşluk içermemelidir.
- XML büyük küçük harf duyarlıdır. Bu nedenle elemanlar açıldıkları şekilde kapatılmalıdır.

Eleman tipleri aynı zamanda genel tanımlayıcı (Generic Identifier-GI) olarak da adlandırılır. Bunun nedeni elemanların DTD veya Şemalarda tanımlanıyor olması ve her eleman yaratıldığında, elemanın tipinde bir örnek (instance) yaratılıyor olmasıdır.

Elemanların genişleyebilme özelliği vardır. Şekil 2.1.'de görüldüğü gibi iç içe geçen elemanlar tanımlayarak XML dökümanı genişletilebilir. Bir XML dökümanında ilk eleman kök eleman olarak adlandırılır ve diğer elemanlar ona bağlı olarak genişler. Şekil 2.1.'de görüldüğü gibi "argument-conf" kök eleman, "argument" elemanlar ise onun çocuklarıdır. "argument-conf" aynı zamanda "argument" elemanların atasıdır. Argument elemanları aynı parent elemana bağlı oldukları için kardeş elemanlardır. Kardeşlik aynı seviyede tanımlanan elemanlar için söz konusudur. Kök(root) olmayan her elemanın sadece 1 tane ebeveyn (parent) elemanı vardır. İki "argument" elemanı altında gördüğümüz "revision" elemanları "argument" elemanlarının çocuğudur. Her bir "revision" elemanının parent'ı farklıdır. Yani ilk "argument" elemanı ile ikinci "argument" elemanı birer parent'tır. Üstteki "argument" elemanı altındaki "revision" elemanı ile alttaki "argument" elemanı içindeki "revision" elemanı kardeş değildir. Ancak her bir <argument> içindeki revision, author, contact çocuk elemanları birbiriyle kardeştir.

Öznitelikler (*Attributes*)

Öznitelikler başlama etiketinde elemanın isminden sonra eklenirler. Bir isim ve değer kısmından oluşurlar ve bir XML dökümanında isteğe bağlı kullanılırlar. Bir

elemanın içinde öznitelik kullanılması dokümanın boyutunu kısaltacaktır. Bu da parser'ın daha hızlı işlem yapmasını sağlayacaktır. Ancak özniteliklerin DTD doğrulaması kolay bir işlem değildir. Eğer öznitelik varsayılan değerli olarak tanımlanmışsa bu parser'ı yoracaktır. Ayrıca öznitelikler iç içe geçmiş şekilde (nested) tanımlanamazlar. Öznitelikler #DEFAULT, #REQUIRED veya #FIXED tanımlanabilir.

Beyaz Boşluk (*White Space*)

XML, satır başı (\r or ch(13)), yeni satır (\n or ch(10)), tab(\t), ve boşluk (' ') karakterlerini white space olarak tanır. Boşluk karakterleri önemli (significant) ve önemsiz (insignificant) olarak ikiye ayırabiliriz. Önemli boşluk karakterleri dokümanın bir parçası olarak görülürler ve saklanırlar. Önemsiz boşluk karakterleri ise XML dokümanının daha anlaşılır olması için kullanılırlar. Parser bu tür boşlukları ihmal eder ve metni düz bir string şeklinde okur. Ancak "xml:space" attribute'ı kullanılarak parser'ın bu boşlukları ihmal edip etmeyeceğini belirtebiliriz. "xml:space" öznitelik bilgisi kök elmandan çocuk elemana devralınır.

İsim Alanları (*Namespaces*)

XML dökümanlarında yaratılan eleman ve özniteliklerin aynı isimde olma olasılığı oldukça yüksektir. Bu da parser'ın belgeyi çözümleyememesi ile sonuçlanır. Bu sorunun çözümlenmesi için XML dokümanlarına teklik sağlayan isim alanı deklarasyonu eklenir. Namespace deklarasyonu "xmlns" öneki ile başlar ve tanımlanan özniteliğin bir XML isim alanı deklarasyonu olduğunu belirtir.

<OReilly:Books xmlns:OReilly="http://www.oreilly.com/">

Burada "xmnls" string'i namespace deklarasyonunun başladığını belirtmekte, "=" işaretinden sonra ise de eleman ya da öznitelik isimleri olarak kullanılacak olan Uniform Resource Identifier(URI) tanımlanmaktadır. Tekliğin sağlanması amacı ile URI'lerin Uniform Resource Locator - URL olması önerilmektedir. URL'lerin fiziksel olarak var olup olmaması önemli değildir.

Varlık (*Entity*)

Varlıklar XML dökümanlarına dışarıdan döküman eklemek ya da sık tekrarlayan verileri tanımlamak için kullanılmakadır.

İyi Biçimlendirilmiş (*Well-Formed*) XML Dökümanı

Bir XML dokümanını özel XML editörle ya da notepad kullanarak yaratabilme esnekliğine sahip olmamıza rağmen, bazı kurallara uyulması gerekir. Parser'ın hatayla karşılaşmaması için dikkat edimesi gereken kurallar şu şekildedir.

1. XML dökümanı en az 1 eleman içermelidir.
2. Bir XML dökümanı sadece bir kök elemana sahip olabilir.
3. Kök olmayan elemanlar başlama etiketi ile başlamalı ve bitiş etiketi ile de bitmelidir.
4. Her bir elemanın 1 ebeveyni olmalıdır.
5. XML büyük küçük harf duyarlı olduğu için elemanlar buna dikkat edilerek yaratılmalı ve iç içe geçen elemanlar örtüşmeyecek (overlap) şekilde tanımlanmalıdır.

Geçerli (*Valid*) XML Dokümanı

Geçerli bir XML dökümanı öncelikle iyi biçimlendirilmiş olmalı ve DTD veya Şema dökümanında belirtilen kurallara uymalıdır. Örnek olarak, DTD dokümanında tanımlanan bir elemanı XML dökümanında kullanmamışsak bu doküman iyi biçimlendirilmiş olabilir ama geçerli değildir.

Belge Türü Tanımı (Document Type Definition –DTD)

DTD'ler XML dökümanı yaratılırken uyulması gereken kısıtların tanımlandığı yapılardır. Bir DTD dökümanı eleman tipi deklarasyonu, öznitelik deklarasyonu, varlık deklarasyonu, notasyon deklarasyonu ve diğer bilgileri içerir. Deklarasyon başlama belirteci (<) ile başlar ve onu büyük harfle tanımlanması gereken anahtar kelime eder. Başlama belirteci ve anahtar kelime kombinasyonu deklarasyon

belirteci olarak adlandırılır. DTD'ler XML söz diziminden farklıdırlar ve XML'in gelişim sürecinde onda da gelişmeler olmuştur. DTD'leri metamarkup dillerin yaratılmasında kullandığımız için metamarkup formlar olarak da düşünebiliriz.

XML Şema

Bir XML dokümanının geçerli olup olmadığını DTD dokümanına veya XML dokümanında kullanılan tanımlamalara uyulup uyulmadığına göre belirleyebiliriz. Ancak XML Şemanın DTD'ye göre birçok avantajı vardır. Bir XML şemanın XML formatında yazılıyor olması, DTD'den daha fazla veri tipini desteklemesi bu avantajlardan birkaç tanesi olarak sayılabilir. Şema elemanları basit ve kompleks olmak üzere ikiye ayrılmaktadır. Basit tip elemanlar alt elemanı olmayan basit veri(string, number, date… v.b.) içeren yapılardır. Kompleks tip elemanlar ise çocuk eleman ya da öznitelik içeren yapılardır.

XML Parser

XML'in en önemli özelliği de veri transferleri için uygun bir yapı sunmasıdır. Bu veri iletimi sırasında uygulamaların XML içinden ihtiyaç duydukları verileri almaları XML parserlar ile olmaktadır. XML Parserlar bir XML dokümanını ya ağaç yapısında (Document Object Model - DOM) ya da olay bazlı tanımlayarak (Simple Api For XML-SAX) üzerinde işlem yaparlar.

Ağaç yapısında işlem yapan DOM parserlarda XML dokümanı içerisinde bulunan tüm elemanlar ağaç yapısında karşılık bulmaktadır. DOM yapısında işlenen doküman memory'ye alındıktan sonra parse edilirken, SAX'ta sadece ihtiyaç duyulan kısım dokümandan alınmaktadır. Her ne kadar DOM'da XML dokümanı belleğe alındıktan sonra üzerinde işlem yapılması büyük dokümanlarda performans sorununa yol açsa da, SAX kodları daha karmaşık ve büyük boyutlu dokümanlar için avantajlıdır. SAX W3C tarafından geliştirilmemiş olsa da DOM için SAX'ın bazı özelliklerinden esinlenilmiştir. SAX java için geliştirilmiş olsa da, diğer platformlarda da java arşiv kütüphaneleri yüklenerek kullanılabilmektedir.

2.2.2. Open GeoSpatial Consortium - OGC

Open GeoSpatial Consortium-OGC 25 Eylül 1994'te kurulan, 400'e yakın şirket, devlet kuruluşu ve üniversiteden oluşan, farklı uygulamaların birlikte çalışabilmesini (interoperability) sağlayacak arayüz standartları geliştiren uluslararası bir organizasyondur. OpenGIS ise, OGC tarafından geliştirilen standartlara ilişkin OGC'nin tescilli bir markasıdır. OpenGIS standartları ile farklı platformlar arasında ideal seviyede uyumlu çalışma, tanımlanan açık standart dokümanları kullanılarak sağlanmaktadır. GML, WFS, WCS ve WMS standartları OGC tarafından geliştirilen open-interface standartlarından birkaçıdır.

WEB Servisleri

Günümüzde farklı işletim sistemleri ve uygulamaların bir arada sorunsuz bir şekilde çalışması önemlidir. Interoperability olarak adlandırılan bu kavram için geliştirilen web servisleri en önemli yapılardan biridir. Günümüzde kullandığımız birçok web sitesinde web servisleri kullanılmaktadır.

XML teknolojisinin gelişmesiyle, daha önce Distributed Component Object Model-DCOM, Common Object Request Broken Architecture-CORBA ve Java Remote Method Invocation-RMI ile yapılması amaçlanan interoperability artık SOAP ile mümkün olmaktadır. SOAP'tan önce sistemlerin belirli işletim sistemleri ya da intranet için daha güvenilir olması SOAP ile büyük oranda çözülmüştür. SOAP, XML temelli bir protokol olup diğer XML tabanlı Universal Description, Discovery and Integration - UDDI, Web Services Description Language - WSDL teknolojileri ile W3C tarafından sunulan standartlar doğrultusunda gelişmeye devam etmektedir.

İstemciden gelen istekler http üzerinden iletilen SOAP mesajları (envelope) ile alınmakta ve yine sunucudan kullanıcıya hızlı ve güvenli bir şekilde SOAP mesajları şeklinde cevap dönülmektedir. UDDI teknolojisi için ise web servislerinin sarı sayfaları demek yanlış olmayacaktır. Web servisleri bu yapılar ile bulunabilmektedir. Kendisi de bir web servis olan WSDL ile uzaktaki makinaların web servisine ulaşılması için gerekli tanımlamalar yapılmaktadır.

SOAP ve WSDL, XML temellerinde, W3C tarafından geliştirilmeye devam edilmekte olup, UDDI W3C'nin bir parçası olmayıp birçok şirket tarafından desteklenerek gelişmeye devam etmektedir. Gerek yerel bir dosyadan gerekse de herhangi bir İlişkisel Veritabanı Yönetim Sisteminde bulunan verilere erişerek bunları web servise çeviren bir takım ticari veya açık kaynak yazılımları bulunmaktadır.

Web Map Service - WMS

WMS, OpenGIS tarafından tanımlanan standarlara sahip, coğrafi verilerin yayınlanması için kullanılan bir web servisidir. WMS de veriler yaygın olarak png, jpeg gibi resim formatında alınır. Resim formatındaki katman istenirse transparan gösterilerek, çok katmanlı gösterimler için avantaj sağlanmış olunur. Verilerin resim formatında gösterilmesi esneklik sağlasa da, verinin farklı boyutlarda istenmesi cache hit'lerinin azalmasına neden olabilmektedir. Bu sorun için tile özelliği bir çözüm olmaktadır.

Web Feature Service - WFS

WFS'i GML formatındaki verinin yayınlanmasını sağlayan bir web servisi olarak söylemek mümkündür. WFS ile GML veri sorgulanabilmekte, update, delete, insert işlemleri yapılabilmektedir. İstemci isteklerini SOAP mesajları olarak gönderip, cevapları da yine SOAP mesajları olarak almaktadır.

WFS'de veri formatı olarak kullanılan GML aşağıda verilmektedir.

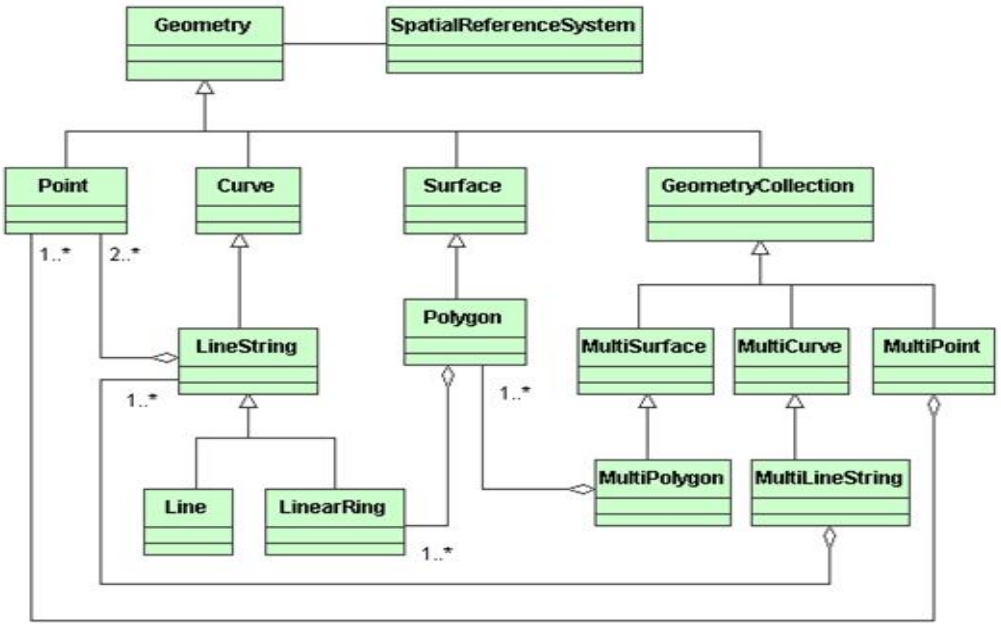
Geography Markup Language - GML

"OGC (Open GIS Consortium) tarafından geliştirilen GML (GML 2001–2004); XML şema tanımına göre coğrafi varlıkların, geometrik ve geometrik olmayan özelliklerine ait bilgilerin modellenmesi, depolanması ve iletilmesini sağlayan bir XML imlemesidir. Dünyayı modellemek için kullanılan GML, OGC'nin ve ISO 19100 serisinin standartlarını temel almaktadır. GML coğrafyayı tanımlamak için obje türlerinden; varlıklar, koordinat referans sistemleri, geometri, topoloji, ölçü

birimleri ve genelleştirilmiş değerler gibi değişik türleri kullanmaktadır. GML aşağıdaki hususları sağlayan XML şeması sözdizimini, mekanizmasını ve anlaşmasını tarif etmektedir (GML, 2001–2004) "[32]

- Coğrafi uygulama şema ve objelerin tanımlaması için açık, satıcı yansız bir çerçeve model sağlamak (opensource),
- Farklı sistemler arası sorunsuz işbirliği sağlamak (interoperability),
- Özel grup ve bilgi birlikleri için, coğrafi uygulama şemalarının tanımlanmasını desteklemek,
- Bağlanılan coğrafi uygulama şemaları ve veri gruplarının bakımı ve yaratılmasını sağlamak,
- Aplikasyon şemaları ve veri gruplarının depolanması ve iletilmesini desteklemek,
- Tanımladıkları coğrafi aplikasyon şemaları ve bilgileri paylaşan organizasyonların kabiliyetlerini artırmak.

Bir GML Şemasını oluşturan Geometri modelini gösteren UML şeması Şekil 2.2'de gösterilmiştir.



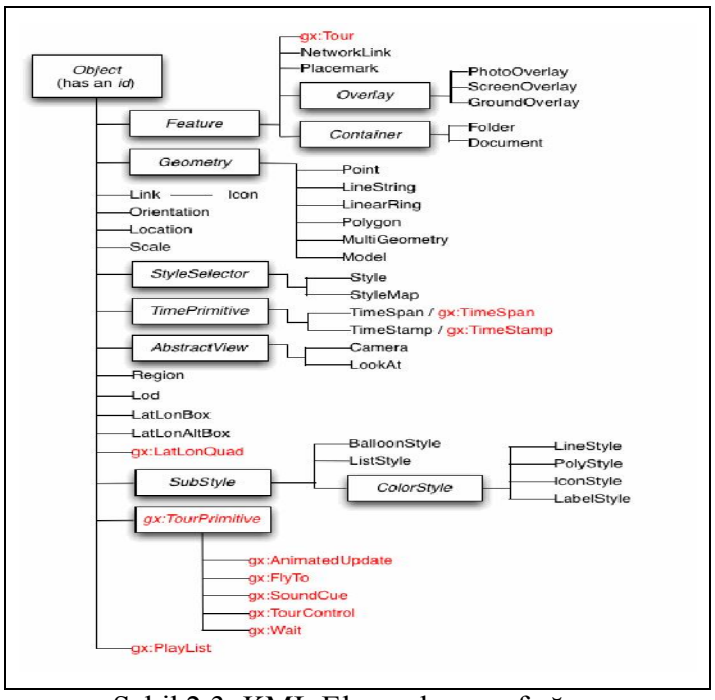
Şekil 2.2. GML geometri modelini gösteren UML şeması [43]

Keyhole Markup Language - KML

KML(Keyhole Markup Language) coğrafi verilerin görüntülenmesi için kullanılan XML tabanlı veri formatıdır. İlk olarak 2001 yılında oluşturulmuş, zaman içerisinde geliştirilmiş ve OGC tarafından onaylanan uluslararası açık standart olmuştur. Resmi olarak da OpenGis KML 2.2 kodlama standartı olarak isimlendirilmektedir. Başlangıçta coğrafi verinin Google Earth üzerinde gösterilmesi için tasarlanmış olsa da web üzerinden coğrafi veri paylaşımı için çok yaygın olarak kullanılmaktadır. KML'nin gösterimi Google Earth, Google Maps, NASA WoldWind gibi gibi birçok uygulama üzerinden sağlanabilmektedir. KML dosyası hem coğrafi veri hem de resim verisi içerebilmektedir. Bu verilerin sıkıştırılarak kullanılabildiği KMZ formatı da web sunucu üzerinden kolaylıkla paylaşılabilmektedir.

KML Dosya Yapısı

Şekil 2.3.'de gösterildiği gibi KML, kutular içinde gösterilen gerçekte KML dosyası içinde kullanılmayan soyut elemanlardan oluşan bir yapıya sahiptir. Bu yapıya göre Placemark elemanı, Feature altındaki bütün elemanları içerir.



Şekil 2.3. KML Elemanları sınıf ağacı

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Document>
<Placemark>
<LineString> <tessellate>true</tessellate>
<altitudeMode>clampedToGround</altitudeMode>
<coordinates>-135,30,500000 -80,30,500000</coordinates>
</LineString>
</Placemark> </Document>
</kml>
```

Şekil 2.4. Örnek KML dökümanı

Şekil 2.4'de olduğu gibi her KML dosyası görünen ilk 2 satır ile başlamaktadır. Bir KML dosyası sadece bir adet <kml> elemanına sahiptir. XML tabanlı bir veri formatı olması sebebiyle KML'de de açılan her tag kapatılmalıdır.

En yaygın kullanılan Google Earth featurelarından biri de Placemark'dır. En basit Placemark, içinde Point elemanı bulundurmaktadır. Placemarklar içinde Point elemanı haricinde LineString, Polygon gibi geometri tipleri tanımlamak da mümkündür. Placemark elemanları <name>, <description> ve <Point>(en temel eleman) şeklinde üç adet elemana sahip olabilir.

<name> çocuk elemanı placemark için etiket olarak tanımlanmaktadır. <description> elemanı placemark hakkında ek bilgiler içermekte olup, bilgi balonu ile ilgili tanımlamalar bu alanda yapılmaktadır. XML dökümanları için de kullanılan CDATA gibi birçok özellik KML dökümanları için de kullanılmaktadır. Overlay'ler ise alttaki tabakanın en üstüne eklenen saydam ya da yarı saydam olabilen resimlerdir. Ground Overlay, yerkürenin kavisli yapısına göre de şekil alan, yerküre zemini üzerine asılı bir şekilde yerleştirilen resimlerdir. Yüksek çözünürlüklü uydu resimleri, yerküre üzerindeki hava durumu yapısını gösteren resimler olabilmektedir. Birçok Earth browser BMP, DDS, GIF, JPG, PGM, 32-Bit PNG, PPM, TGA ve TIFF forfatını desteklemektedir. Screen, Photo ve Network şeklinde biçimleri vardır.

KML Formatındaki Verilerin Gösterimi

KML veriyi Google Earth kullanarak göstermek mümkün olduğu gibi, şu anda Microsoft Virtual Earth, NASA World Wind, ESRI ArcGIS Explorer, Google Maps, Google Maps for Mobile ya da Yahoo! Pipes ile oluşturulan bir çok uygulama tarafından desteklenmektedir. Bu tez kapsamında Google Earth ve Google Maps kullanılmış olup, ilerleyen bölümlerde detaylı bilgi verilmiştir.

Web Coverage Service-WCS

"Web Raster Servisi (Web Coverage Service WCS) mevcut veriyi detaylı tanımlamaları ile birlikte sağlar. Bu verilere karşılık gelen karmaşık sorgulamalar yapılmasına olanak verir ve sadece resmedilmiş değil yorumlanabilir ve sonuç çıkartılabilir bir veriyi orjinal semantiği (resimler yerine) ile geri gönderir. Bu haliyle, gerçek vektör veriyi döndüren Web Feature Sevice (WFS) ve sayısal bir görüntü dosyası üreten Web Map Service (WMS)'den farklıdır" [31].

2.2.3. Harita WEB Servislerini Hazırlamak için Kullanılan Sunucu Yazılımları

ArcGIS Server

"ArcGIS Server sunucu-tabanlı komple ve bütünleşmiş bir coğrafi bilgi sistemidir. Mekânsal veri yönetimi, görselleştirme ve mekansal analize yönelik kullanıma hazır son kullanıcı uygulamalarını içerir. ArcGIS Server, CBS kullanıcılarının kendi masaüstü bilgisayarlarından 2D ve 3D haritaları ve coğrafi analizleri yönetmelerini ve bunları bütünleşmiş araçlarla ArcGIS Server üzerinden yayınlamalarını sağlayan gelişmiş bir CBS sunucusudur" [37].

ArcGIS Server

- Browser-tabanlı erişim,
- Diğer kurumsal sistemlerle uyum (CRM, ERP),
- CBS ve IT birlikte işletilebilirlik standartlarını destekler,
- .Net veya Java ile özel uygulamalar geliştirebilme.

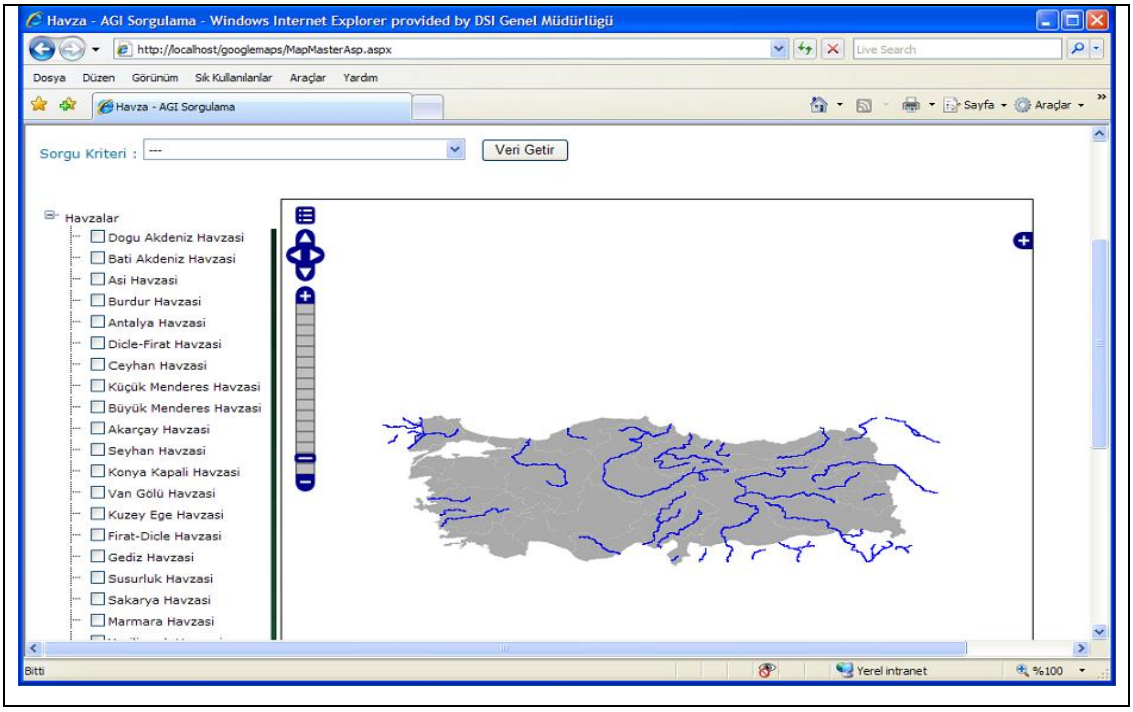
Özelliklerine sahiptir.

MapServer

MapServer açık kaynak kodlu, coğrafi verinin web üzerinde gösterimine imkan veren, PHP, Python, Perl, Ruby, Java ve. NET gibi scripting dilleri ve yazılımlarını destekleyen bir geliştirme ortamıdır. OGC tarafından tanımlanmış standartları desteklemekte; ESRI shapfiles, PostGIS, ESRI ArcSDE, Oracle Spatial, MySQL veri formatlarını da destekleyen, kullanımı sürekli artan bir uygulamadır.

GeoServer

Geoserver coğrafi verilerin görüntülenmesini sağlayan açık kaynak kodlu Java tabanlı sunucu yazılımıdır. Geoserver ile verileri; Google Maps, Google Earth, Yahoo Maps, and Microsoft Virtual Earth gibi harita gösterim uygulamalarının birçoğu üzerinde görüntülemek mümkün olmaktadır. Aşağıda Havzalar ve Nehirler katmanlarının Geoserver üzerinden WMS servisi ile yayımlanmasına ait şekil görülmektedir.



Şekil 2.5. Havzalar ve Nehirler katmanlarının Geoserver üzerinde gösterimi

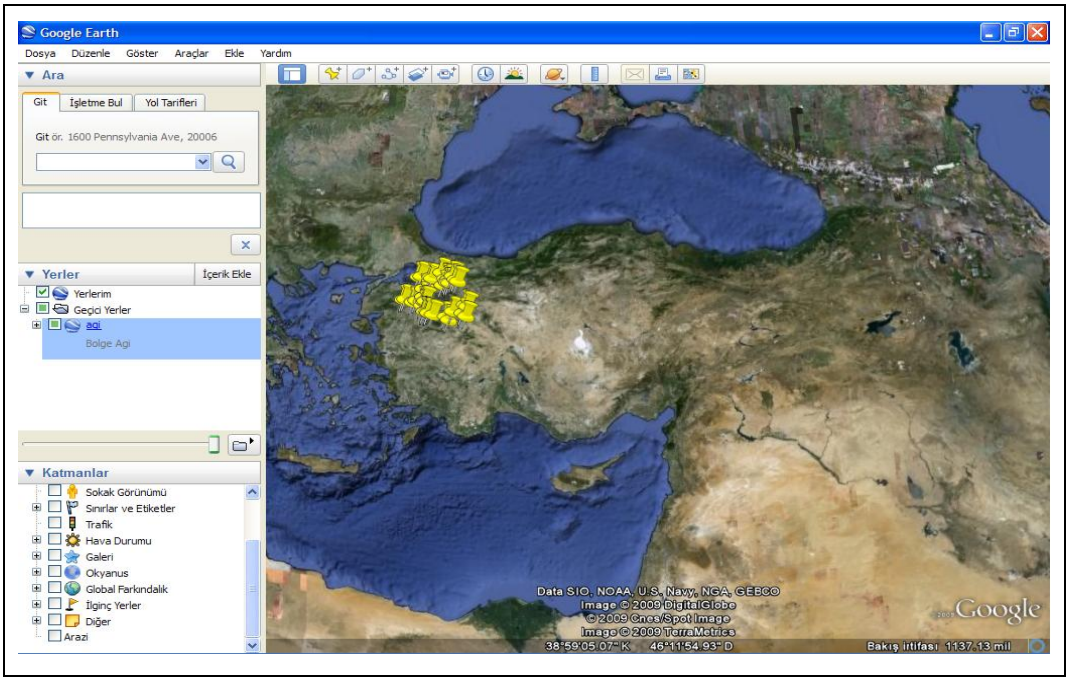
2.2.4. Halihazırda Kullanılan Harita WEB Servisleri

Google Earth

Google Earth, Earth viewer adı ile Keyhole firması tarafından geliştirilmiş olan, 2004 yılında Google tarafından firmanın satın alınması ile Google Earth olarak isim değiştiren, yerkürenin sokak, bina, insan derinliğine kadar görüntülenebildiği, hatta uzaydaki galaksilere kadar gezilebildiği ücretsiz bir yazılımdır. Google Earth vasıtasıyla görüntülenen veriler internet yoluyla güncellenmektedir.

Google Earth vasıtası ile objeleri 3 boyutlu görüntülemek mümkün olmaktadır. Google Earth Sayısal Yükseklik Modeli (Digital Elevation Model -DEM) NASA tarafından yürütülen Shuttle Radar Topography Mission-SRTM projesi kapsamında sayısal yükseklik haritaları ile üretilen yeryüzünün topografik haritalarının görüntülenmesini sağlayan bir yazılımdır. Bu sayede yeryüzünün 3 boyutlu görüntülenmesi mümkün olmaktadır. 3 boyutlu modellenen bina, köprü gibi yapılar KML dosyaları içine alınarak Google Earth üzerinden gösterilmektedir.

Google SketchUp uygulaması ile 3 boyutlu modeller oluşturmak ve bunları Google Earth vasıtasyla dünya ile paylaşmak mümkün olmaktadır. Bu tez kapsamında KML dosyası olarak kaydedilen iki boyutlu Akım Gözlem İstasyonları, Havza ve Nehir verileri Şekil 2.5.'de de görüldüğü gibi Google Earth üzerinde gösterilmiştir.



Şekil 2.6. Google Earth üzerinde örnek Akım Gözlem İstasyonları gösterimi

Google Maps

Google Maps, web sunucusu üzerinden, üretilen coğrafi verilerin görüntülemesini sağlayan bir servistir. Bu servis sayesinde pahalı uydu görüntülerini ve haritalarını satın almadan internet bağlantısı ile kullanmak mümkün olmaktadır.

Google Maps, Google Maps API'si ile web sayfasına gömülerek kullanılmaktadır. Google Maps API'sini kullanabilmek için öncelikle API key'i aşağıda görüldüğü yapıda alınarak web sayfasına eklenmelidir.

http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAzr2EBOXUKnm jVnk0OJI7xSosDVG8KKPE1-m51RBrvYughuyMxQ-i1QfUnH94QxWIa6N4U6MouMmBA

Bu tez kapsamında uygulama içine gömülen Google Maps Api'si ile kullanılan altlık üzerinde KML dosyası olarak üretilmiş olan Havza, Nehir gibi coğrafi veriler de görüntülenmiştir.

Virtual Earth

Virtual Earth, Microsoft'un yeni nesil haritalama ve lokasyon hizmetidir. Bu MapPoint Web Hizmeti kuşbakışı uydu ve hava görüntüleri, harita stilleri gibi yenilikleri, gelişmiş yerel arama etrafında birleştirmektedir. Virtual Earth platformu sayesinde, şirketler, tüketicilere işletme yeri, veriler ve yerel bilgileri kolayca arama, görselleştirme imkânı sağlayan bir yapı oluşturmaktadır.

Yahoo Maps

Yahoo tarafından sunulan harita görüntüleme servisidir. Sunulan Api'ler sayesinde haritaları web uygulaması içine gömerek kullanmak mümkün olmaktadır.

Yukarıda en çok kullanılan harita servisleri verilmiş olup bunun dışında ilave web servisleri bulunmaktadır. Bu çalışma kapsamında yaygın olarak kullanılmakta olan "Google Maps" harita web servisi altlık haritalar için kullanılacaktır.

2.3. Sunum Katmanında Kullanılan Mimari Altyapı

Buraya kadar olan kısımda;

- Cografi verilerin "İlişkisel Veritabanı Yönetim Sisteminde" depolanması ve yönetilmesi,
- "İlişkisel Veritabanı Yönetim Sisteminde" saklanan verilere dışarıdan ulaşılabilmesi için web servislerine dönüştürülmesi veya Sunum katmanında kullanılacak programlar tarafından okunabilmesini sağlayacak formata dönüştürülmesi,

amacıyla yararlanılan mevcut teknoloji altyapı anlatılmıştır. Bu kısımda ise mevcut coğrafi verilerin web ortamında sunulmasını sağlayacak teknoloji altyapısı hakkında kısaca bilgi verilecektir.

Verilerin paylaşımı, tanımlanması için çalışmalar aslında 1960'lı yıllara dayanmaktadır. İlk defa William W. Tunnicliffe tarafından 1967'de dillendirilen dökümandan içerik ve formatın ayrılması fikri generic coding'in ilk aşaması

olmuştur. Çalışmalar sonucunda Text Description Language adı verilen dil ilk olarak geliştirilmiş, bu dil 1971'de Generalized Markup Language(GML) adını almıştır ve orijinal doküman değiştirilmeden içeriğe ulaşılabilinmiştir.

Bundan sonra birçok çalışma yapılmış ve 1986'da Standard Generalized Markup Language-SGML (ISO 8879) tanımlayıcı işaretleme dili tanımlanabilmesini sağlayan uluslararası bir metalanguage standartı olarak onaylanmıştır. SGML bundan sonra birçok metalanguage için temel olmuş, HTML ve XML de ondan türetilmiştir. SGML ve SGML temelli dillerde veri yapısı, content ve stil ayrılabilmiş, başlangıçta GML ile ulaşılmak istenen amaca bir nebze ulaşılabilinmiştir.

Ardından aynı zamanda world wide web'in de çıkış yeri olan the European Organization for Nuclear Research(CERN) 'de 1990'lı yılların başında SGML bazlı Zengin Metin İşaret Dili (HyperText Markup Language – HTML) geliştirilmiştir. SGML'e göre basit ve anlaşılır olması sevindirici olsa da semantik ve verinin yapısı sebebiyle, gelecekteki büyük ölçekli uygulamalar için daha gelişmiş bir dil kulanımı ihtiyacı ortaya çıkmıştır. World Wide Web Consortium'un da katkılarıyla SGML ekibi, HTML kadar anlaşılır ama ondan daha gelişmiş bir dil için çalışmalara başlanmış ve SGML'in çok da kritik olmayan, kullanılmayan kısımları da çıkartılarak Extensible Markup Language 1.0 (XML 1.0) W3C tarafından uygun bulunmuştur. Böylece platform ve software bağımsız XML oluşmuştur. World Wide Web Consortium HTML ve XML in birleştirilmesi ile oluşan Extensible HyperText Markup Language - XHTML'i 2000 yılında onaylamıştır. XHTML, HTML dökümanının XML yazım kurallarına uygun olarak üretilmesi ile oluşturulmaktadır. Bu da stil ile verininin ayrıştırılmasını sağlayarak, farklı web tarayıcılarda gösterilen verilerin her ne kadar yazım hatası yapılmış olsa da bazı tarayıcılarda gösterilebilen sayfaların, farklı platformlarda gösterilememe sorununu çözecektir.

2.3.1. JavaScript Nedir, JavaScript Kütüphanelerinin Kullanımı

Javascript Internet Explorer, Firefox, Opera ve Safari gibi tüm büyük tarayıcılar için internet üzerinden çalışan en popüler betik dilidir. HTML sayfalarına etkileşim eklemek için tasarlanmıştır. "Netscape, Inc. tarafından geliştirilen JavaScript

programlama dili, Java platformunun bir parçası değildir. JavaScript programlama dilinin Java programlama diliyle bazı ortak özellikleri vardır ama bu dilden ayrı olarak geliştirilmiştir"[39]. JavaScript hem bir nesne yönelimli dil hem de prosedürel olarak işlev görebilmektedir. JavaScript dinamik bir betik dilidir. Javascript kodları HTML dökümanı içerisine script etiketleri arasında eklenerek ya da JavaScript dosyaları HTML dosyası içinden çağrılmak sureti ile çalıştırılabilmektedir.

Javascript kütüphaneleri javascript temelli uygulamaların kolay geliştirilmesini sağlamaktadır. AJAX API Kütüphanesi en popüler açık kaynak kodlu JavaScript kütüphanesidir. Bu çalışmada, hazırladığımız web sayfalarında dinamik olarak harita bileşenlerini kullanmak ve halihazırda internette yayımlanmakta olan coğrafi veri servislerine ulaşabilmek amacıyla değişik JavaScript kütüphaneleri hazırlanmıştır. "API" olarak hazırlanan bu JavaScript kütüphanelerini kullanarak sayfamıza rahatlıkla "Google Maps", "Virtual Earth", "Yahoo Map", "Open Street Map" gibi uygulamalardan uydu görüntüleri veya yol, akarsu, göl gibi coğrafi veri setlerini içeren bileşenleri ekleyerek, bu veri setleri üzerinde değişik analizler yapılarak sonuçlar hazırlanan web sayfalarında görüntülenebilmektedir. En çok kullanılan iki "JavaScript Kütüphanesi" aşağıda anlatılmaktadır.

Google Maps API

Google Maps JavaScript API, web sayfalarına Google Maps'in gömülmesini sağlamaktadır. API kullanmak için, ilk olarak "http://code.google.com/intl/tr/apis/maps/signup.html" adresinden kayıt olmak suretiyle bir API anahtarı edinmek gerekmektedir. Bir API anahtarı alındığında harita uygulaması geliştirebilmektedir. Bu yolla Google maps üzerinden haritalara açık bir şekilde ulaşmak mümkün olsa da, Api açık kaynak kodlu değildir.

Openlayers API

OpenLayers API, herhangi bir web sayfasına kolaylıkla dinamik bir harita koyma imkânı sağlamaktadır. OpenLayers, dünya çapında birçok organizasyon tarafından geliştirilmiş ve desteklenmiştir. Web tarayıcılarında harita verisi görüntülemek için kullanılan tamamen açık kaynak kodlu bir JavaScript kütüphanesidir. OpenLayers

coğrafi veri erişimi için OpenGIS Consortium standartlarından Web Map Service (WMS) and Web Feature Service (WFS) protokollerini destekler. Openlayers Api'si kullanarak Google maps haritalarını görüntülemek mümkün olmaktadır.

Bu çalışmamızda, internette mevcut olan tüm harita web servislerine erişilebilen ve aynı zamanda gerek dosya tabanlı olarak (KML ve XML formatında) gerekse de web servis olarak hazırlanan coğrafi verilerimizi birlikte kullanma imkânı sunan "Openlayers API" kullanılmıştır[46].

BÖLÜM 3

ORACLE SPATIAL BİLEŞENİNİN YAPISI

Bu çalışma kapsamında daha önce de belirtildiği gibi mevcut verilerin İlişkisel Veritabanı Yönetim Sisteminde depolanması ve yönetimini sağlamak üzere Oracle Spatial kullanılmaktadır. Bu bölümde,

- Oracle Spatial'da coğrafi verilerin yaratılması, sorgulanması için kullanılan komutlar,
- Oracle Spatial'da coğrafi verilerin yaratılmasında kullanılan obje türleri,
- Oracle Spatial'da veriye erişim için kullanılan indeksleme yapısı

gibi temel bileşenler ayrıntılı olarak anlatılmıştır. Ravi Kothuri, Albert Godfrind, Euro Beinat tarafından kaleme alınan "Pro Oracle Spatial" kitabı da Oracle tarafından sunulan dokümanlarla birlikte bu çalışmaya önemli katkı sağlamıştır.

3.1. SDO_GEOMETRY Object tipi

SDO_GEOMETRY veri tipi Oracle'ın point, linestring, polygon ve birçok karmaşık geometrilerinden oluşan Spatial verilerin saklandığı nesne tipidir. Oracle SDO_GEOMETRY nesne tipi Şekil 3.1.'de görüldüğü şekilde tanımlanmaktadır.

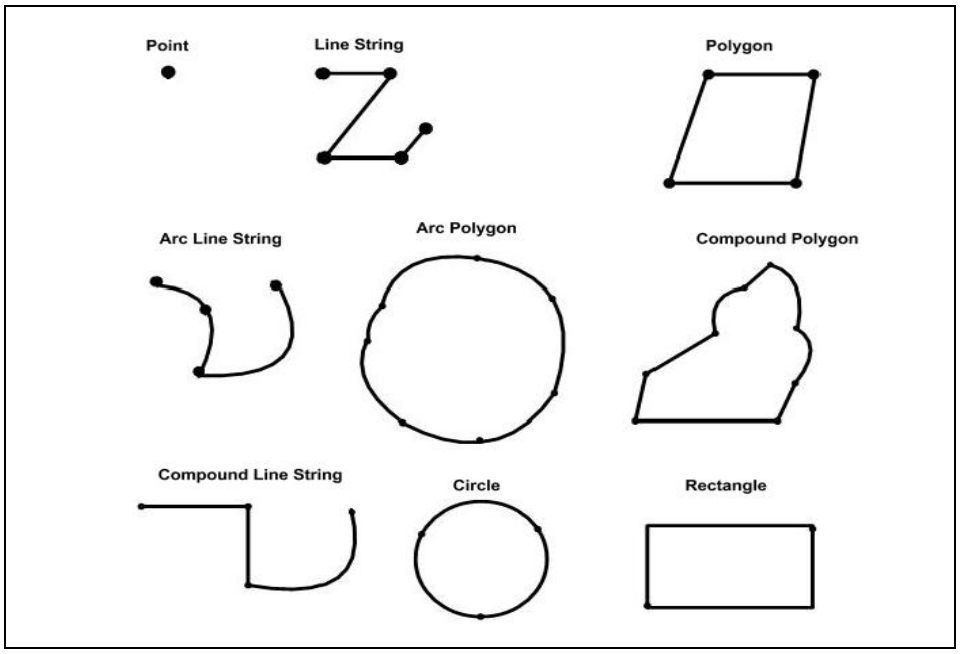
```
CREATE TYPE sdo_geometry AS OBJECT
(
        SDO_GTYPE NUMBER,
        SDO_SRID NUMBER ,
        SDO_POINT SDO_POINT_TYPE,
        SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
        SDO_ORDINATES SDO_ORDINATE_ARRAY
);
```

Şekil 3.1. Oracle SDO_GEOMETRY nesne tipi

SDO_GEOMETRY kolonu tablo yaratılırken Şekil 3.2.'de belirtildiği şekilde eklenebilmektedir.

```
CREATE TABLE geometry_examples
(
   name VARCHAR2(100),
   description VARCHAR2(100),
   geom SDO_GEOMETRY
);
```

Şekil 3.2. Geometri kolonuna Sahip Tablonun Yaratılması



Şekil 3.3. Geometri Tipleri

Şekil 3.3.'de Oracle tarafından kullanılan geometry tipleri görülmektedir. Spatial veri modeli hiyerarşik olarak, elemanlar, geometriler ve layer'lardan oluşmaktadır.

Temel spatial elemanları point, linestring ve poligonlar'dır. Elemanları şehir ya da ülke sınırları, nehirler şeklinde ifade etmek mümkündür.

Geometrileri ise, temel elemanların seti olarak düşünebiliriz. Bir geometri, homojen ya da heterojen elemanlar topluluğu olabilir.

Layerlar elemanlardan oluşan geometriler topluluğudur. Aynı karakteristiğe sahip geometrilerin bir araya getirilmesi ile oluşmaktadır. Örneğin Havza, Nehir gibi katmanlar tanımlanabilmektedir.

```
CREATE TYPE sdo_geometry AS OBJECT
(
   SDO_GTYPE NUMBER,SDO_SRID NUMBER,SDO_POINT
   SDO_POINT_TYPE,SDO_ELEM_INFO
   SDO_ELEM_INFO_ARRAY,SDO_ORDINATES SDO_ORDINATE_ARRAY
);
```

Şekil 3.4. SDO_GEOMETRY veri tipi

Şekil 3.4.'de gösterilen SDO_GEOMETRY veri yapısını incelediğimizde;

SDO_GTYPE ile geometrinin tipi belirtilmektedir. Geçerli geometri tipleri "Geometry Object Model for the OGIS Simple Features for SQL Specification" dökümanında belirtilmiş olan geometry tipleridir. SDO_GTYPE özelliği 4 basamaklı "dltt" formatındaki değerle ifade edilir. Bu formatta;

"d" verinin boyutunu göstermektedir. 2, 3 ya da 4 olabilir.

"l" Lineer Reference Sisteminde ölçülen boyutu göstermektedir. Linear Referencing System geometrisi olmayan verilerde ki bu tezde kullanılan verilerde olduğu gibi, bu değer 0'dır.

"tt" 00-07 aralığında değerler alabilir. 08-99 arası değerler ilerideki çalışmalar için ayrılmıştır. Çizelge 3.1.'de geçerli SDO_GTYPE değerleri gösterilmiştir.

Çizelge 3.1. Geçerli SDO_GTYPE değerleri

Değer	Geometri	Açıklama
00	UNKNOWN_GEOMETRY	Spatial bu değeri ihmal eder
01	POINT	Bir Point elemanı
02	LINE or CURVE	Curve, lineer veya her ikisinide içerebilen line string elemanı
03	POLYGON	Poligon elemanı
04	COLLECTION	Heterojen elemanlar topluluğu
05	MULTIPOINT	Bir veya daha fazla nokta içeren eleman
06	MULTILINE or MULTICURVE	Bir veya daha fazla line string içeren eleman
07	MULTIPOLYGON	disjoint olabilen çoklu poligon içeren eleman

Bu veriler ışığında SDO_GTYPE'ın "2001" olarak tanımlanması, 2 boyutlu point verisini ifade etmektedir.

SDO_SRID kolonu ile kullanılan koordinat sistemi belirtilmektedir. Tabloda kullanılacak SRID değeri SDO_COORD_REF_SYS tablosundan alınır ve USER_SDO_GEOM_METADATA view tablosunda belirtilen SRID ile aynı olmalıdır.

SDO_POINT özelliği ise point türü verileri tanımlamak için kullanılmaktadır. SDO_POINT türü veriler tanımlanırken SDO_ELEM_INFO ve SDO_ORDINATES'e "NULL" değeri atanır. Şekil 3.5.'de AGI tablosuna point bir verinin eklenmesi için kullanılan SQL cümlesi görülmektedir.

```
INSERT INTO AGI(OBJECTID, GEOM)  VALUES
(
   1266, SDO_GEOMETRY(2001,8307,
   SDO_POINT_TYPE(43.309390000,39.068950000, NULL),NULL, NULL)
);
```

Şekil 3.5. AGI tablosuna veri eklenmesi

```
SELECT geom FROM havzalar hvz where hvz.OBJECTID=1;

GEOM

----------------------------------------

(2003, 8307, , (1, 3, 1), (34.899366, …, 34.649343, 36.80604,
34.649334, 36.8051, 34.649198, 36.804475))
```

Şekil 3.6. Havzalar tablosundan veri sorgulama

Şekil 3.6.'da poligon tipinde tanımlanmış bir havza verisi görülmektedir. Bu ifade incelendiğinde;

SDO_ELEM_INFO, SDO_ORDINATE, içerisinde tanımlanan koordinat verilerinin yorumlanmasına yardımcı olan tanımlamalardır. SDO_ELEM_INFO içerisinde veriler 3'er 3'er tanımlanmaktadır. Eğer geometri 1 elemana sahipse, SDO_ELEM_INFO değeri { 1,1003,1} şeklinde 3 sayı ile ifade edilir. Belirtilen bu üç değerin anlamlarını incelediğimizde;

İlk değer SDO_STARTING_OFFSET olarak ifade edilmektedir ve burada SDO_GEOMETRY.SDO_ORDINATES(1) şeklinde de ifade edilebilir.

İkinci olarak SDO_ETYPE tanımlanır. SDO_ETYPE elemanın tipini belirtmektedir. SDO_ETYPE'ın 1003 ya da 2003 olması, geometrinin dış (1) ya da iç (2) olmasını göstermektedir.

Üçüncü değer SDO_INTERPRETATION'dır. SDO_ETYPE'ın bileşik (compound) eleman olup olmamasına bağlı olarak 2 tür değer alabilir. Eğer SDO_ETYPE compound elemansa 4 ya da 5 değerini alır. Eğer SDO_ETYPE compound eleman değilse, bu interpretation değeri elemanın ordinat'ların seti olarak yorumlanabilir.

SDO_ETYPE ve SDO_INTERPRETATION değerlerinin nasıl yorumlanacağı birkaç örnek veri üzerinden Çizelge 3.2.'de verilmektedir.

SDO_ORDINATES attribute'ı ise değişken uzunluklu array tipinde tanımlanan, geometrinin içerdiği elemanlara ait bütün boyutlarda koordinat verilerinin saklandığı alandır.

Çizelge 3.2. SDO_ETYPE ve SDO_INTERPRETATION değerleri [3]

SDO_ETYPE	**SDO_INTERPRETATION**	**Açıklama**
0	Herhangi nümerik değer	Type 0 eleman.Oracle Spatial tarafından desteklenmeyen tipleri ifade eder.
1	1	Point Tipi
1	9	Yönlü point tipi
1003 or 2003	1	Kenarları Düz Line'lar şeklinde birbirine bağlı poligon tipi veri. Kapalı bir poligon için Son Koordinat değeri Başlangış Koordinat değeri ile aynı olmak zorunda.
1003 or 2003	2	Dairesel bir şekilde yay'lar ile birbirine bağlı poligon. Poligondaki ilk yayın başlangıç noktası ile son yayın bitiş noktası aynı noktayı göstermelidir.

3.2. İndeks Yapısı

Mekânsal verilerin çok boyutlu olması sebebi ile tek boyutlu normal index yapısının kullanımı bu tür verilerde uygun olmamaktadır. Çok büyük boyutta mekansal veriler üzerinde sorgulama yapıldığında, doğru index kullanımı verimli sorgulamaların yapılmasında büyük etkendir. Oracle, spatial veriler üzerinde R-tree ve Q-tree index kullanımını destekler. Önemli olan, bu indeksleme mekanizmalarının hangisinin yapılacak çalışma için uygun olduğudur. "Oracle Spatial User's Guide and Reference" dökümanında R-tree indeks kullanılması salık veriliyor olsa da bazı durumlarda q-tree indeks kullanımı daha uygun olmaktadır.

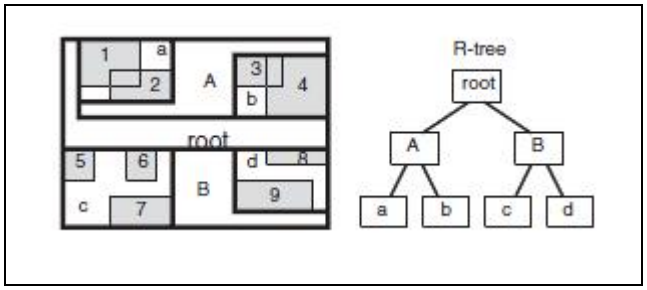
R-tree ve Q-tree indeks arasındaki temel farklar aşağıdaki gibidir:

1. Q-tree yapıda geometriler, her biri aynı boyuttaki tile'lar ile ifade edilirken, R-tree yapıda Minumum Bounding Rectangle'lar ile tanımlanmaktadır. Tile yapısı ile q-tree, join işlemlerinde çok başarılı olmaktadır.
2. R-tree indexlerin yaratılması ve tune edilmesi daha kolayken, Q-tree indexlerin tune edilmesi ciddi performans sorunlarına neden olmaktadır.
3. R-tree indexler için ayrılan alan, Q-tree indexler için ayrılan alandan çok daha küçüktür.
4. Özellikle belirlenen geometri için En Yakın Komşuları bulma sorgusu R-tree index yapılarında çok daha hızlıdır.
5. Çok fazla güncelleme işlemleri yapıldığında R-tree ağaç yapısında bozulmalar olabilmekte ve performansın iyileştirilmesi için index'in yeniden düzenlenmesi gerekirken, Q-tree index performansı etkilenmemektedir.
6. R-tree index mekanizmasının Q-tree 'den önemli bir farkı da R-tree de 4-boyutluya kadar veri indexleyebilirken, Q-tree de bu 2-boyutlu veriye kadar sınırlanmıştır.
7. Spatial sorgulamalarının önemli bir parçasını oluşturan SDO_WITHIN_DISTANCE sorgularında R-tree index önerilmektedir.
8. R-tree bütün yerkürenin indexlenmesi için uygunken Q-tree uygun değildir.

Bu sebeplerden dolayı da çalışmamızda R-tree indeks kullanımı tercih edilmiştir.

3.3. R-tree İndeks Yapısı

R-tree indeksler B-tree indeks yapısından türetilmiş, spatial veriler için yapılan performans testlerinde büyük oranda başarılı olmuş, en yaygın kullanılan algoritmadır. Mekânsal veriler için K-D-B ve Quad Tree algoritmalarının pek de verimli olmadığı gözlenmiş, K-D-B Tree algoritmasının daha çok Point veriler için uygun olduğu görülmüştür[24].



Şekil 3.7. R-tree indeks yapısı [3]

Şekil 3.7.'de görüldüğü gibi 1,2,3,4,5,6,7,8 ve 9 içerdikleri geometrilere ait Minimum Bounding Rectangle-MBR'leri ifade etmektedir. A ve B disk page olarak da ifade edilen, sorguların performansı açısından da sınırlı sayıda MBR içerebilen node'lardır. Node'lar en az Maximum içerebilecekleri geometri sayısının (M) yarısı kadar (m<=M/2) geometri içermelidir. a, b, c ve d ise leaf node'lar olmakla birlikte veritabanındaki Rowid ve MBR değeri tutulmaktadır. Non-leaf node'larda ise child node'a işaret eden işaretçi(pointer) ve child nodları cover eden MBR'leri belirtilir. R-tree indeksler üzerinde yapılan çalışmalar sonucunda R-tree indekslerin bir varyantı olan ve yaklaşık %30 daha performanslı çalışan R*-tree indeksler geliştirilmiştir. R*-tree indeksler veri yapısı olarak R-tree indekslerle aynı olmakla birlikte, yeni bir geometri eklendiğinde node'lar için maximum MBR sayısı aşılmışsa (owerflow) bazı geometrileri ağaçtan çıkartıp yeniden ekleyerek (forced-reinsertion algorithm) ağaç yapısının yeniden düzenlemesi ile daha geliştirilmiş yapılardır. Mekânsal sorguların hızlandırılması için çalışmalar sürdürülmekte olup, Oracle tarafından R*-tree yapısı veritabanına uygulanmıştır.

Oracle R-tree indeks ağaç yapısını veritabanında "MDRT" ile başlayan veritabanı tablolarında tutmaktadır. Bu tablolar veritabanında başka bir yere taşınmamalı ya da değiştirilmemelidir. Tablo export edildiğinde, bu tabloların ayrıca yedeklerinin alınmasına gerek yoktur. Tablo import işlemi esnasında da spatial indeksler düzgün bir şekilde yaratılacaktır.

Mekânsal veriler üzerinde yapılacak yoğun update işlemleri sonucunda bu ağaç yapısında bozulmalar olabilmekte bu da sorguların performansında düşüşlere neden olmaktadır. Bu durumda Oracle'ın sunduğu

SDO_TUNE.QUALITY_DEGREDATION fonksiyonu sayesinde R-tree ağaç yapısının tekrar yapılandırılmasına gerek olup olmadığı belirlenebilmektedir. Eğer bu fonksiyonun sonucu 2'den büyük çıkıyorsa indeksin yeniden yapılandırılması düşünülmelidir. Bu fonksiyonun formatı Şekil 3.8.'de gösterilmektedir.

```
SDO_TUNE.QUALITY_DEGRADATION(
     schemaname IN VARCHAR2,
     indexname IN VARCHAR2
     ) RETURN NUMBER;
Ya da
SDO_TUNE.QUALITY_DEGRADATION(
     schemaname IN VARCHAR2,
     indexname IN VARCHAR2,
     indextable IN VARCHAR2
     ) RETURN NUMBER;
```

Şekil 3.8. SDO_TUNE.QUALITY_DEGREDATION fonksiyonu yapısı

3.4. R-tree İndeks Yaratılması

R-tree indeksler, mekansal analizlerin yapılabilmesi ve veriye hızlı erişim için tablo üzerinde bulunan SDO_GEOMETRY veri tipindeki kolon üzerinde yaratılmalıdır. İndeks yaratılmadan önce USER_SDO_GEOM_METADATA görüntü(view) üzerinde, katmana ait bilgilerin Şekil 3.9.'da belirtildiği şekilde eklenmesi gerekmektedir.

```
INSERT INTO USER_SDO_GEOM_METADATA
(TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES       ('AGI',       'GEOM',              MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X',      -180.000000000,      180.000000000,
0.500000000),           MDSYS.SDO_DIM_ELEMENT('Y',  -90.000000000,
90.000000000, 0.500000000)       ),   8307);
```

Şekil 3.9. AGI tablosuna ait ilgili bilgilerin USER_SDO_GEOM_METADATA view'ına eklenmesi

Bu işlemin ardından Şekil 3.10.'da görüldüğü gibi Spatial index yaratılmalıdır.

```
CREATE INDEX agi_spatial_idx ON AGI(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Şekil 3.10. Spatial index yaratılması

İndeksin yaratılması sırasında bir problem olursa ve indeks bir şekilde fail olursa "DROP INDEX agi_spatial_idx" örnek SQL komutunda olduğu şekilde index drop edilmelidir.

İndeks yaratma işlemi sırasında dikkat edilmesi gereken bazı değerlerden bahsetmek gerekirse;

Kullanılan bütün indekslerde olduğu gibi R-tree indexler de veritabanında belirli bir alan kaplar ve "Data Manipulation Language-DML" işlemleri ile birlikte indexler de değişikliğe uğrar. Oracle'ın spatial veriler için sunduğu SDO_TUNE paketi içinde gelen estimate_rtree_index_size fonksiyonu ile index yaratılırken ne kadar alana ihtiyaç duyulacağı belirlenebilmektedir. Bu fonksiyon ile index yaratıldıktan sonra gerekli alanı megabyte düzeyinde vermektedir. İndeks yaratılma sırasında tablespace'de bu değerin 2-3 katı kadar boş alan olması gerekmektedir. Şekil 3.11.'de bu fonksiyonun kullanımı görülmektedir.

```
SELECT sdo_tune.estimate_rtree_index_size
(
   'SPATIALNDX', -- şema adı
   'AGI', -- tablo adı
   'GEOM' – spatial index bulunan kolon adı
) sz
FROM dual;

SZ
----------
1
```

Şekil 3.11.İndeks yaratılırken gerekli alan için "Sdo_tune.estimate_rtree_index_size" fonksiyonunun kullanımı

Ayrıca yüksek bellek ihtiyacı gerektiren indeks yaratma işleminde bir veritabanı parametresi olan SORT_AREA_SIZE'ın Şekil 3.12.'de görüldüğü gibi 1MB'a set edilmesi indeks yaratılırken performans artışı sağlamaktadır.

```
1. ALTER SESSION SET parameter_name = xxx;
2. EXECUTE SYS.DBMS_SYSTEM.SET_INT_PARAM_IN_SESSION
(999, 8888, 'sort_area_size', 1000000);
```

Şekil 3.12. "sort_area_size" parametresinin set edilmesi

Oracle Saptial indeks yaratılırken kullanılabilen bazı parametreler sorgu performansını artırmaktadır. Bu parametreleri aşağıdaki şekilde açıklayabiliriz.

3.4.1. LAYER_GTYPE Parametresi

Bu parametre ile geometrinin tanımlı olduğu SDO_GEOMETRY kolonunda tanımlanabilecek geometri tipi sınırlanmaktadır. Böylece sorgu performansı artırılabilmektedir. Geçerli sdo_gtype değerleri için Layer_gtype parametresini kullanmak mümkün olmaktadır. Bu parametrenin kullanımı Şekil 3.13.'de görülmektedir.

```
CREATE INDEX AGI_SPATIAL_IDX ON AGI(GEOM)

INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS ('LAYER_GTYPE=POINT');
```

Şekil 3.13. AGI tablosu üzerinde LAYER_GTYPE parametresi kullanılarak mekansal indeks yaratılması

3.4.2. SDO_INDX_DIMS Parametresi

Bu parametre ile mekansal indeksin boyutu belirtilmektedir. Bu parametrenin değeri varsayılan olarak 2'dir. Bu da çok boyutlu verilerde sadece verininin ilk 2 boyutunun indexlenmesine sebep olur. Eğer çok boyutlu verilerde çalışılıyorsa verininin istenilen kadar boyutu indekslenerek ek işlemler önlenmiş olmaktadır.

```
CREATE INDEX AGI_SPATIAL_IDX ON AGI(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS ('SDO_INDX_DIMS=2');
```

Şekil 3.14. AGI tablosu üzerinde SDO_INDX_DIMS parametresi kullanılarak spatial index yaratılması

3.4.3. SDO_DML_BATCH_SIZE Parametresi

Veriler üzerinde yapılan değişikliklerden sadece verinin kendisi değil veriye ait indeks de etkilenir. Mekânsal veri üzerinde ekleme, güncelleme ve silme işlemleri fazlaca yapılıyorsa, SDO_DML_BATCH_SIZE parametresi ile tek defada güncellenen indeks sayısı artırılarak performans iyileştirilebilir. Bu parametre için indeks yaratılırken bir değer atanmamışsa, default değeri 1000'dir. Şekil 3.15.'de bu parametrenin kullanımı görülmektedir.

```
CREATE INDEX AGI_SPATIAL_IDX ON AGI(GEOM)INDEXTYPE IS
MDSYS.SPATIAL_INDEX PARAMETERS ('SDO_DML_BATCH_SIZE=5000');
```

Şekil 3.15. AGI tablosu üzerinde SDO_DML_BATCH_SIZE parametresi kullanılarak spatial index yaratılması

3.5. Spatial Operatör, Prosedür ve Fonksiyonlar

Oracle Spatial PL/SQL API'si içinde birçok Operatör, Prosedür ve Fonksiyon barındırmaktadır. Operatörler, mekansal veriler üzerinde analizlerin yapılabilmesi için prosedür ve fonksiyonlara göre sorguların daha yüksek performanslı çalıştırılmasını sağlamaktadır. Operatörlerin prosedür ve fonksiyonlardan en önemli farkı sorgulara hız kazandıran mekansal indekslerin kullanılabiliyor olmasıdır. Prosedür ve fonksiyonlarda ise mekansal indeks tanımlansa dahi kullanılmamaktadır. Bu nedenle sorguları çalıştırırken operatörler ile yapılabilecek işlemlerde, prosedür ve fonksiyonları tercih etmemek daha doğru olacaktır. Örneğin SDO_RELATE operatörü yeterli ise SDO_GEOM.RELATE'i kullanmamak daha uygun olmaktadır. Spatial operatörleri WHERE cümleciğinde kullanırken, SDO_GEOM, SDO_CS, and SDO_LRS gibi prosedür ve fonksiyonları hem WHERE cümleciğinde hem de subquery'lerde kullanmak mümkün olmaktadır. Başlıca Spatial Operatörler

SDO_FILTER, SDO_JOIN, SDO_NN, SDO_NN_DISTANCE, SDO_RELATE ve SDO_WITHIN_DISTANCE'dır.

Şekil 3.16.'da görünen Spatial Operatörlerin sözdizimini incelediğimizde; table_geometry, operatörün uygulanacağı geometri kolonunu ifade etmekte, query_geometry sorgu lokasyonunu göstermektedir. parameter_string mekansal operatörler için tanımlanan parametreleri göstermekte ve opsiyonal'dir.

```
<spatial_operator>
(
 table_geometry IN SDO_GEOMETRY,
 query_geometry IN SDO_GEOMETRY
 ' <some_parameter>'
) = 'TRUE'
```

Şekil 3.16.Spatial Operatör yapısı

Yaygın Kullanılan Spatial Operatörlere kısaca bakacak olursak;

3.5.1. SDO_WITHIN_DISTANCE Operatörü

SDO_WITHIN_DISTANCE operatörü, query_geometry parametresi ile tanımlı geometri için, belirtilen uzaklık içerisinde kalan geometrilerin saptanması amacıyla kullanılmaktadır. Bu operatörün çalıştırılabilmesi için kullanılan geometri kolonları üzerinde mekansal indeks tanımlanmış olmalıdır. İki boyutun üstündeki geometrik veriler için bu operatör kullanılamamaktadır. SDO_WITHIN_DISTANCE operatörü ile birincil filtre ya da birincil ve ikincil filtrenin birlikte çalıştırılması "querytpe" parametresinin set edilip edilmemesi ile mümkün olabilmekterdir. "Querytype=FILTER" şeklinde bir tanımlama yapılmadıysa varsayılan olarak birincil ve ikincil filtre kullanılmaktadır.

3.5.2. SDO_NN Operatörü

Bu operatör belirli bir mesafe içerisinde bulunan geometrileri bulmak yerine, en yakın geometrileri bulmaya odaklanmıştır. Çevre geometrilerin uzaklıklarının tahmin edilemediği durumlarda ideal bir çözüm olmaktadır. SDO_NN operatörünün çalışabilmesi için "table_geometry" kolonu için mekansal indeksin yaratılmış olması gerekmektedir. Ayrıca indekslenen 2 boyut üstündeki verilerde de bu operatör çalışmamaktadır.

Operator yapısına göre incelediğimizde; table_geometry mekansal indeksin kullanıldığı geometry kolunu, query_geometry sorgu lokasyonu olmaktadır. Param parametresi 2 adet performans paremetresi alabilmektedir. Bunlar, sdo_batch_size ve sdo_num_res parametreleridir.

3.5.3. SDO_JOIN Operatörü

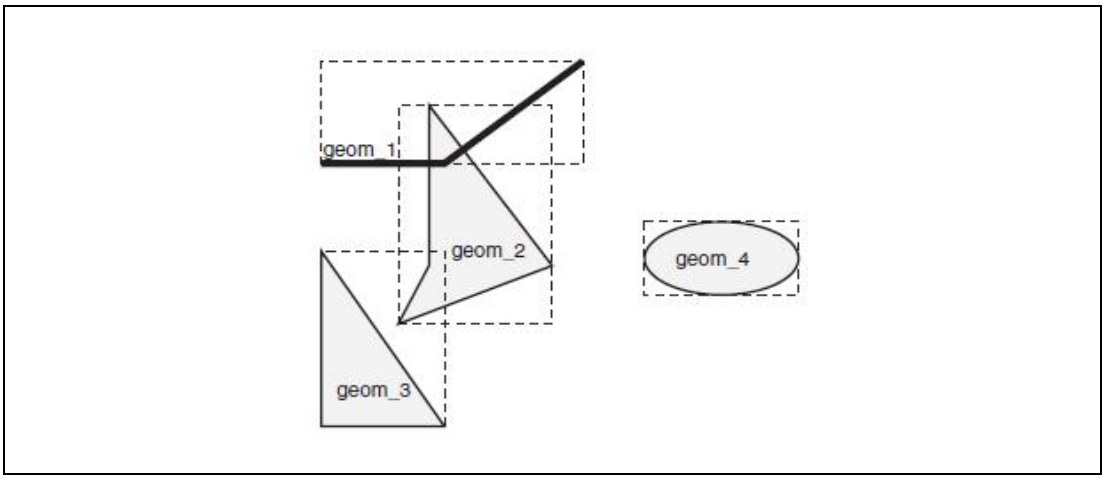
```
SDO_JOIN
(
      table1 IN VARCHAR2,
      col1 IN VARCHAR2,
      table2 IN VARCHAR2,
      col2 IN VARCHAR2
      [, parameter_string IN VARCHAR2
      [, preserve_join_order IN NUMBER]]
)
```

Şekil 3.17. SDO_JOIN yapısı

Belirli bir mesafe içinde kalan ve where cümleciğinde ayrıca istenen ölçüt olduğunda SDO_WITHIN_DISTANCE operatörünün kullanımı daha uygun olmaktadır. Ancak where cümleciğinde bir kısıtlama yoksa ve bir tablonun bütün satırları için diğer tablonun bütün satırları çalıştırılmak isteniyorsa SDO_JOIN operatörü daha yüksek performans göstermesi sebebiyle daha uygun olmaktadır.

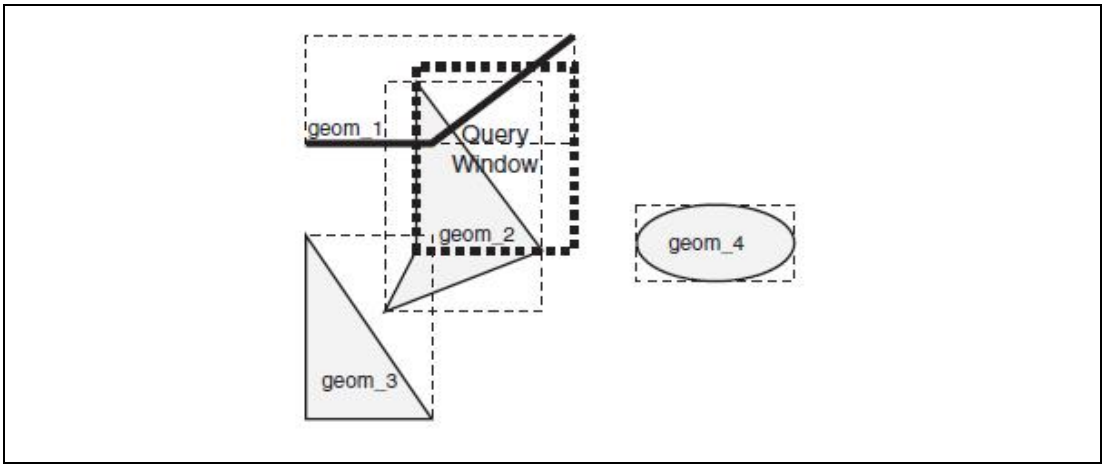
3.6. Sorgulama Modeli ve Sorguların Değerlendirilmesi

Spatial R-tree indeks yapısında her bir geometri "Minumum Bounding Rectangle-MBR" ile ifade edilir. Şekil 3.18.'de görüldüğü gibi her bir geometri için MBR'ler kesik çizgi ile ifade edilmektedir.



Şekil 3.18.Tanımlı geometrilere ait MBR'ler [2]

Sorgulama yapıldığında Şekil 3.19.'da görüldüğü gibi kalın noktalı çizgi ile ifade edilen sorgu penceresinin (query window) dokunduğu MBR'ler ya da direkt olarak dokunduğu geometriler değerlendirilerek analiz yapılır. MBR'ler spatial indeks tablosunda tutulmaktadır.



Şekil 3.19. Tanımlı geometriler için kullanılan Query window[2]

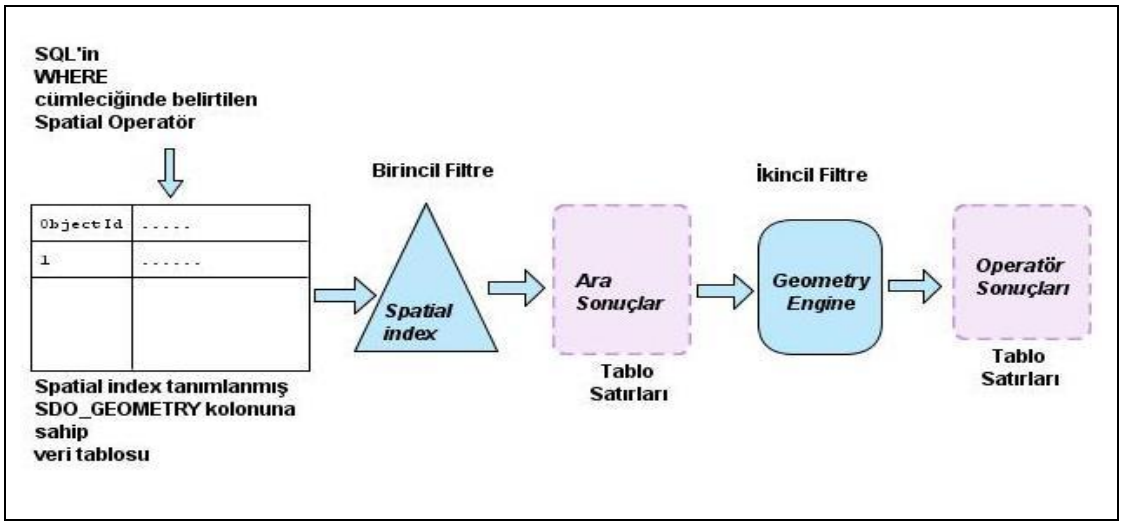
3.6.1. Birincil Filtre (*Primary Filter*)

Mekânsal indeks kullanımı ile operatörlerin değerlendirilmesi birincil filtre olarak adlandırılır. Bu kullanıma SDO_FILTER operatörü örnek olarak verilebilir. MBR'lerin dokunduğu geometrilerle işlem yapılarak sonuca ulaşılır. Sorgu değerlendirilirken direkt geometri üzerinde işlem yapılmadığı için yaklaşık sonuç elde edilir.

3.6.2. İkincil Filtre (*Secondary Filter*)

İkincil filtre operatörlerine SDO_RELATE örnek olarak verilebilir. Bu tür sorgularda primary filter sonucunda oluşturulan veri seti ikincil filtre işlemine tabi tutulur ve direkt olarak geometriler üzerinde işlem yapılarak kesin sonuca ulaşılır.

Birincil ve İkincil Filtre Mekanizması Şekil 3.20.'de görülmektedir.



Şekil 3.20. Oracle Spatial sorgu değerlendirme mekanizması

3.7. Verilerin Veritabanına Aktarılması

Oracle SDO_GEOMETRY veri tipi, mekansal verilerin diğer metin verileri ile birlikte tabloda tutulabilmesi için aşağıda görüldüğü gibi tablo yaratılırken oluşturulabilmektedir.

```
CREATE TABLE AGI(OBJECTID NUMBER,geom SDO_GEOMETRY);
```

Şekil 3.21.'de metin verilerin sdo_geometry veri tipi ile birlikte veritabanına eklenmesine bir örnek SQL cümlesi görülmektedir.

```
INSERT INTO HAVZALAR VALUES
(
1, -- OBJECTID
SDO_GEOMETRY -- SDO_GEOMETRY
(
2003, -- 2 boyutlu poligon
8307, -- SRID
NULL, -- SDO_POINT_TYPE
SDO_ELEM_INFO_ARRAY (1, 1003, 1), -- Sadece 1 ring içeren poligon
SDO_ORDINATE_ARRAY -- SDO_ORDINATES
(
34.899154, 36.7406, -- ilk vertex koordinatları
 34.894715, 36.748573, -- diğer vertex'ler
34.888716, 36.756454,34.884808, 36.760521,
34.880699, 36.764105,34.879165, 36.765997,
34.875175, 36.768745,34.871309, 36.771073,
34.866809, 36.77466, 34.859859, 36.779624,
34.854705, 36.782902,
34.848636, 36.78577 – son vertex)));
```

Şekil 3.21. Havzalar tablosuna coğrafi verilerin eklenmesi

Verilerin fazla olduğunu düşünürsek veritabanına toplu olarak aktarılması daha uygundur. Bu amaçla bu tez kapsamında kullanılan veriler veritabanına sql loader ile toplu olarak atılmıştır. Sql Loader verileri SQL insert komutları ile tabloya eklediği için çok büyük veri kümeleri atılırken, hız problemleri yaşanabilmektedir. Bunların önüne Direct-Path Loading metodu kullanılarak geçilebilmektedir[6]. Bu tez kapsamında kullanılan veriler çok büyük boyutlarda olmadığı için Ek A'da görünen prosedürle veritabanına atılmıştır.

Bu tez kapsamında, veriler öncelikle ESRI ürünü olan ArcGIS programı tarafından üretilen shape file formatında alınmış, oracle tarafından shape file'ları oracle SDO_GEOMETRY veri tipine dönüştüren "shp2sdo" uygulaması kullanılmıştır. Sh2sdo'nun çalışması sonucunda üretilen "agi.ctl" ve "agi.sql" dosyaları çalıştırılarak veriler atılmıştır. Shp2sdo.exe'nin çalıştırılması sonucu üretilen "agi.sql" dosyasının içeriği de Ek A'da görülmektedir.

Atılan SDO_GEOMETRY verilerin geçerli olup olmadığı da yine Oracle'ın sunduğu SDO_GEOM paketinin bir parçası olan VALIDATE_GEOMETRY_WITH_CONTEXT ve VALIDATE_LAYER_WITH_CONTEXT fonksiyonları ile belirlenmektedir.

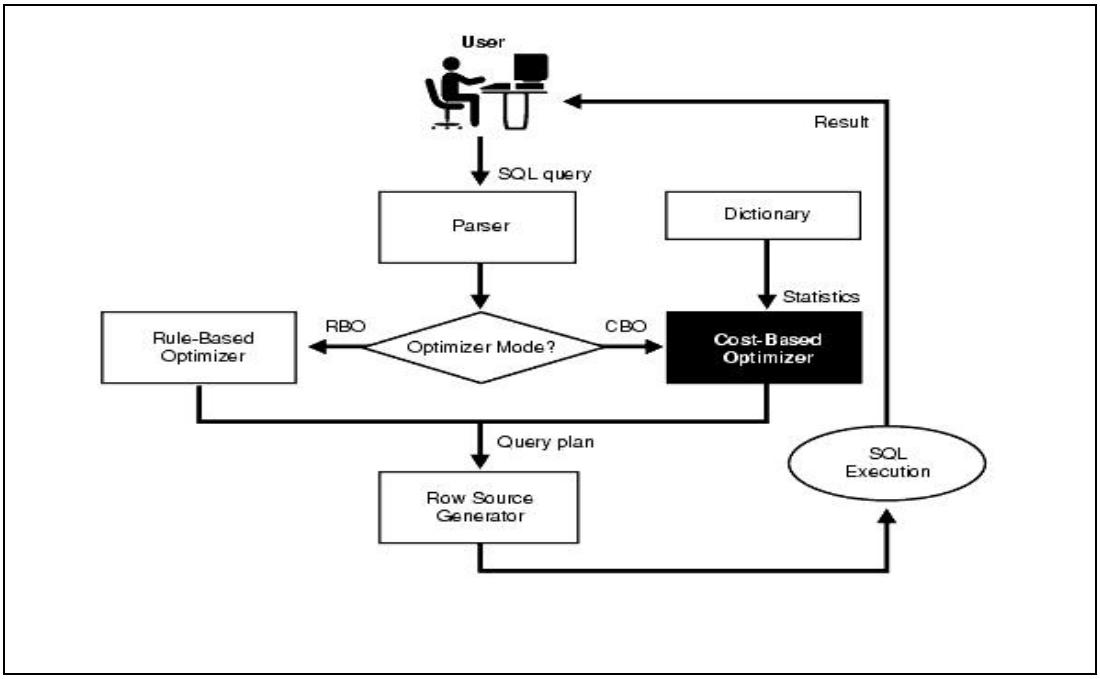
BÖLÜM 4

PERFORMANS DEĞERLENDİRİLMESİ İÇİN KULLANILAN ARAÇLAR

Bu bölümde veritabanı üzerinde yapılan performans testleri için sunulan araçlar, veritabanının sorgu değerlendirme stratejisi anlatılmıştır.

4.1. Veritabanı Performans Değerlendirme

Mekânsal verilerin performans değerlendirmesinin doğru bir şekilde yapılabilmesi için kullanılan, çalıştırılan SQL komutlarının CPU ve I/O kullanımları önemli bir ölçüttür. Bu çalışmada kapsamında geliştirilen uygulama üzerinden çalıştırılan SQL komutlarının CPU ve I/O kullanımları incelenerek performans değerlendirilmesi yapılmıştır. Ayrıca üretilen "Explain Plan"lar, alınan "Automatic Workload Repository-AWR rapor"ları bu değerlendirmede kullanılan başlıca materyallerdir. Veritabanı tasarım aşamasındayken veritabanı parametrelerinin, uygulamanın yapısına göre set edilmesi önemlidir. Ancak bu yapılmamışsa ve geri dönülemiyorsa veritabanının çalışma hızını kötü yönde etkileyen SQL'lerin saptanması ve tune edilmeleri gerekmektedir. Bir SQL cümlesinin çalışması parse edilmesi, optimize edilmesi ve çalıştırılması aşamalarından oluşmaktadır. Şekil 4.1.'de kullanıcı tarafından çalıştırılmak istenen bir SQL cümlesinin işlenmesi genel olarak gösterilmiştir.



Şekil 4.1. SQL cümlelerinin oracle optimizer tarafından değerlendirilmesi [42]

Parse etme işleminde SQL'in sözdizimi ve semantik kontrolü yapılır. Bu aşama System Global Area-SGA'da bulunan library cache'de gerçekleşir.

Optimize edilme aşamasında ise Oracle optimizer'ını kullanarak en optimal veri erişimini sağlamaya çalışır. Cost Based Optimizer-CBO toplanan istatistikleri kullanarak ürettiği çalışma planları arasından en etkin olan çalışma planını seçer.

Çalıştırılma aşamasında ise optimize edilen SQL çalıştırılır.

Bu tez kapsamında, üzerinde en çok duracağımız faz Optimize etme fazıdır. Bu çalışmada mekansal sorgular için optimizasyon yöntemlerinin nasıl kullanacağı, SDO operatörlerinin seçiminin performanstaki önemi, Oracle Spatial teknolojisinin getirdiği performans özelliklerine dikkat çekilecektir.

4.2. Cost-Based Optimizer

Optimizer DML işlemleri için(INSERT, UPDATE, DELETE) en uygun çalışma planını üretmektedir. Bu iterative bir süreçtir. Veritabanı üzerinde çalışan uygulamaların değişmesi, farklı veri setlerine ihtiyaç duyulması gibi sebeplerle

veriye yeni erişim yollarının denenmesi gerekmektedir. Optimizer'ın karar vermesi toplanan veritabanı istatistiklerine bağlıdır. İstatistikler ile tablolar ve kullanılan indeks bilgilerinin alınması yanında sistem istatistiklerinin de toplanması en doğru ve optimal sonuca ulaşmada önemlidir. İstatistiklerin toplanması Oracle 10g veritabanında GATHER_STATS_JOB'ı ile otomatik olarak yapılmaktadır. Ancak bazen manuel olarak da istatistik toplanması gerekmektedir. Bu durumda DBMS_STATS paketi kullanılarak object ve sistem istatistikleri toplanabilmektedir. Ayrıca Oracle Enterprise Manager-OEM kullanarak da istatistik toplamak mümkün olmaktadır. Bu şekilde optimizer ile en uygun çalışma planları üretilmektedir. Ancak bazı durumlarda, kullanılan istatistiklerin güncel olmaması, Optimizer tarafından seçilen veriye erişim metodunun en uygun çözüm olmamasına neden olmaktadır. Bu durumda çeşitli hint'ler de kullanılarak veritabanı seçilen erişim şeklini kullanmaya zorlanabilir. Bu hint'lerle yeni çalışma planlarının seçilmesi sağlanmış olur. Eğer elde hiç istatistik yoksa oracle yardımcı diğer verilere göre çalışma planına karar verir.

CBO ile aynı zamanda join işlemlerinin gerçekleştirilmesi için Access path(Full Table Scan, Table Access Rowid, Index Scans), Join metod(Nested-Loop, Hash Join, Sort-merge Join) ve Join Order(2 ya da daha fazla tablo varsa hangisinin temel alınacağı) lara karar verilmektedir.

Oracle'da fazla kaynak tüketen SQL'lerin belirlenmesi, bu SQL'lerin optimize edilmesi için önemlidir. Bu da çeşitli şekillerde mümkün olmaktadır. "AWR raporları", "V$SQL view"ı, "Automatic Database Diagnostic Monitor-ADDM" ve "SQL Trace" top SQL'leri bulmada yardımcı olmaktadır.

Optimizasyon için en çok kullanılan SQL performans tuning araçları explain plan, Autotrace and SQL Trace'dir.

4.3. SQL Tuning

SQL tuning için öncelikle en çok kaynak tüketen SQL'lerin tespit edilmesi gerekmektedir. Bu SQL'ler tespit edildikten sonra sorgu optimizasyon araçları (explain plan, Autotrace and SQL Trace) yardımıyla optimizasyon işlemleri yapılabilmektedir.

En çok kaynak tüketen SQL'leri belirlemenin başlıca yolları; AWR raporları, Automatic Database Diagnostic Monitor- ADDM, SQL Tuning Advisor, SQL Access Advisor, StatSpack ve V$SQL view'lardır.

AWR raporları 10g ile gelen, veritabanının anlık veya belirli zaman aralığı içinde snapshot'larının alınarak performans bilgilerinin elde edilmesini sağlamaktadır. SQL'lerin tespiti için uygun bir yöntem olarak kullanılmaktadır.

ADDM, AWR raporlarının güncellenmesi ile güncellenen, bu raporlar doğrultusunda, önerilerle veritabanı yöneticisini yönlendiren uygulamadır. AWR ve ADDM Oracle 10g Enterprise Edition ile gelen ve ayrıca lisanslanan teşhiş paketleridir.

Manuel olarak SQL'lerin tune edilmesi çok zahmetli bir süreçtir. Bu nedenle SQL Tuning Advisor ve SQL Access Advisor Oracle 10g ile gelen önemli tuning paketleridir. Bu paketler ile SQL'ler otomatik olarak analiz edilebilmekte, üretilen çözümler sisteme entegre edilebilmektedir.

StatSpack ise, Oracle8i'den bu yana kullanılabilen, AWR ve ADDM gibi ayrıca lisanslanması gerekmeyen performans teşhis aracıdır. Statspack halen kullanılabilir olmakla birlikte oracle tarafından AWR'nin kullanılması tavsiye edilmektedir.

V$SQL view'ları ise peformansı çok kötü olan SQL'lerin belirlenmesi için kullanılan view'lardır.

Bu tez kapsamında bu tuning paketlerinin ve optimizer araçlarının birçoğu kullanılmış olup, çalıştırılan SQL'lerin CPU ve I/O değerleri alınarak, gerektiğinde tavsiyeler de kullanılarak yeni SQL'ler sisteme uygulanmıştır.

4.4. Sorgu Planı (*Explain Plan*)

Explain plan, Oracle Optimizer tarafından SQL'in çalışması için seçilen çalışma planının tune edilebilmesi için önemli bir araçtır. İstenilen şekilde sorgu yeniden yazılarak Optimizer'ın seçtiği çalışma planı yeniden değerlendirilebilmektedir. Execution plan'ı almak için "Oracle Enterprise Manager -OEM" ve "Tool for Application Developers-TOAD" gibi GUI araçları da kullanılabilmektedir.

Explain Plan üretebilmek için öncelikle plan tablosunun yaratılması gerekmektedir. Bu global ya da şema içinde local olabilir. Birçok şekilde Explain Plan üretmek mümkün olmaktadır. Şekil 4.2.'de plan tablosu yaratılmakta, Şekil 4.3.'de belirtilen sorgu için explain plan oluşturulmaktadır. Yaratılan explain plan da Şekil 4.4.'de belirtilen şekilde görülebilmektedir.

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan.sql-----1x

Table created.
```

Şekil 4.2. Plan tablosunun yaratılması

```
SQL> EXPLAIN PLAN FOR
SELECT * FROM AGI WHERE DURUMU='A'  AND AGI.NEHIR_AD='Aci Çay'

Explained.
```

Şekil 4.3. Bir sorgu için explain plan yaratılması

```
SQL> SELECT * FROM table (DBMS_XPLAN.DISPLAY);
```

Şekil 4.4. Explain plan'ın görüntülenmesi

4.5. Autotrace

Autotrace aracı ise Explain planların otomatik olarak, SQL cümlesi çalıştırılır çalıştırılmaz üretilmesini sağlamaktadır. Autotrace özelliğinin aktif olabilmesi için bir defaya mahsus olmak üzere plan tablosunun yaratılmış olması gerekmektedir. Aşağıda görüldüğü gibi, Autotrace özelliği aktif hale getirdikten sonra SQL cümlesi çalıştırıldığında Explain plan otomatik olarak üretilmektedir. "set autotrace traceonly" komutu çalıştırılarak sadece explain plan'ın görülmesi de sağlanabilmektedir. Autotrace özelliği "set autotrace off" komutu kullanılarak kapatılmadığı sürece de çalıştırılan SQL'ler için explain plan alınmaya devam edilmektedir.

```
SQL> SET AUTOTRACE ON
SQL> SELECT ct.HAVZANO
    FROM havzalar comp, agi ct
    WHERE comp.OBJECTID=1
    AND SDO_RELATE(ct.geom, comp.geom, 'MASK=ANYINTERACT' )='TRUE';

Execution Plan
----------------------------------------------------------
Plan hash value: 2038277376
-------------------------------------------------------------------------------------
| Id | Operation                    | Name           |Rows| Bytes |Cost (%CPU)|
Time    |
-------------------------------------------------------------------------------------
|   0| SELECT STATEMENT             |                | 25 |  4475 |    5 (0)
|00:00:01 |
|   1|  NESTED LOOPS                |                | 25 |  4475 |    5 (0)
|00:00:01 |
|*  2|   TABLE ACCESS FULL          | HAVZALAR       |  1 |    57 |    3 (0)
|00:00:01 |
|   3|   TABLE ACCESS BY INDEX ROWID| AGI            | 25 |  3050 |    5 (0)
|00:00:01 |
|*  4|    DOMAIN INDEX              | AGI_SPATIAL_IDX|    |       |          |
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
   2 - filter("COMP"."OBJECTID"=1)
   4 - access("MDSYS"."SDO_RTREE_RELATE"("CT"."GEOM","COMP"."GEOM",'mask=ANYINTE
RACT              querytype=window  ')='TRUE')
Statistics
----------------------------------------------------------
        548  recursive calls
          4  db block gets
       1578  consistent gets
          5  physical reads
          0  redo size
        850  bytes sent via SQL*Net to client
        407  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
         14  sorts (memory)
          0  sorts (disk)
         32  rows processed
```

Şekil 4.5. Çalıştırılan SQL cümlesi için Explain plan

Şekil 4.5.de görünen explain plan'ı incelediğimizde; çalışma planında görünen en içteki satır ilk önce çalıştırılmaktadır. Ayrıca burada 4.sırada bulunan Domain index AGI_SPATIAL_IDX'in SDO_RELATE operatörü için optimizer tarafından kullanıldığı görülmektedir. Mekânsal indeksler ve diğer non-native indeksler Oracle tarafından Domain index olarak tanımlanmaktadır. Yine çalışma planında görünen 2. ve 3. satırlar aynı seviyede olmaları sebebiyle üstteki satır ilk olarak çalıştırılacaktır. Burada Havzalar tablosuna objectid=1 şartından dolayı full table access yapılmış, AGI tablosuna da Rowid 'lere göre erişim yapılmıştır. Ayrıca iki tablonun satırları arasında nested loop join yapılmıştır. 10.000 satırdan daha küçük tablolar için nested loops join ideal bir çözüm olmakla birlikte, eğer optimizer böyle bir plan çıkarmasa dahi kullanılan hint'ler yardımıyla nested loop'a zorlamak mümkündür.

4.6. SQL Trace and TKPROF

SQL trace, çalıştırılan SQL'lerin izlenmesini sağlamaktadır. Üretilen trace file'larının daha okunaklı olması için ise, TKPROF hizmetinden yararlanılmaktadır.

SQL Trace aracı, sadece explain plan almanın yanında çalıştırılan SQL'lerin CPU ve I/O tüketimi hakkında da bilgi vermektedir. Bu araç ile Oracle'ın internal olarak çalıştırmış olduğu SQL'ler hakkında da detaylı bilgi alınabilmektedir. Örnek bir SQL trace prosedürü ve sonucu Ek E'de görülmektedir.

BÖLÜM 5

ÇALIŞMADA İZLENEN GENEL METODOLOJİ

Bu bölümde, çalışmada kullanılan verilerin özellikleri, hazırlanan web uygulamasının mimari yapısı, veritabanı yapısı, verilerin veritabanına atılması ve uygulamada kullanılan prosedürler detaylı bir şekilde anlatılmıştır.

5.1. Kullanılan Coğrafi Veriler

Bu çalışma kapsamında kullanılan coğrafi veriler "ESRI Shapefile" formatında temin edilmiştir. Söz konusu veriler ilerideki bölümlerde kapsamlı olarak anlatılmakta olup, ilk olarak çalışmada kullanılacak "Oracle SDO_GEOMETRY" formatına "shp2sdo" programı kullanılarak çevirilmiştir. Daha sonra söz konusu veriler "SQL Loader" programı vasıtasıyla Oracle RDBMS ortamına aktarılmıştır. Söz konusu coğrafi veriler Orcale RDBMS ortamında "Oracle Spatial" bileşeni sayesinde depolanabilmektedir.

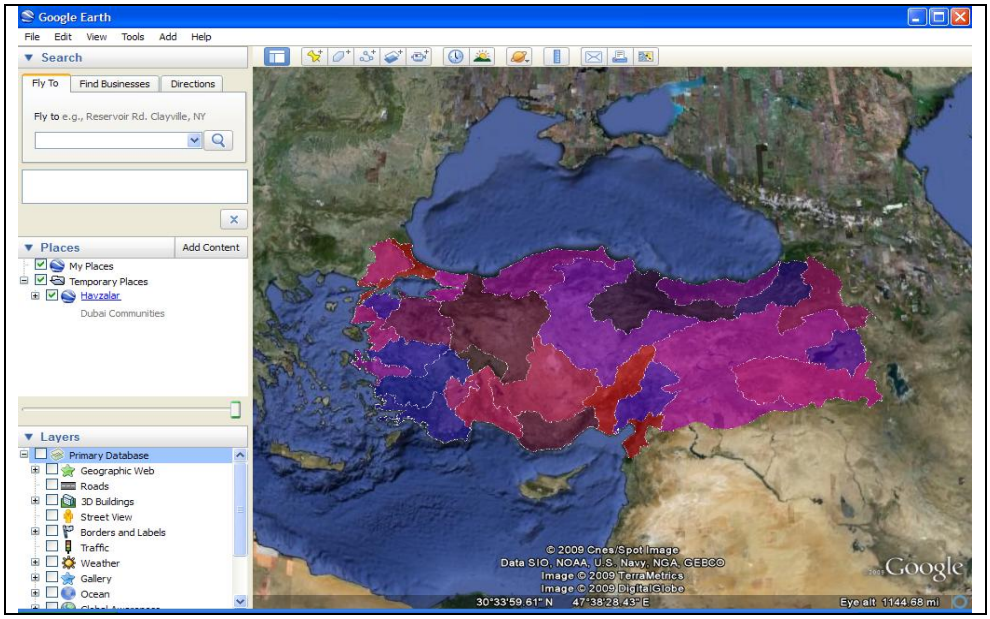
Bu çalışma kapsamında performans testlerinin yapılmasında veya hazırlanılacak uygulama programı ile Havzalar, Akarsular, İller, Göller, Barajlar, Akım Gözlem İstasyonları kullanılarak değişik coğrafi sorgular hazırlanmış, kullanılan coğrafi katmanlar ile de uygulama görsel olarak zenginleştirilmiştir. Çalışmada kullanılan veriler Devlet Su İşleri Genel Müdürlüğü Coğrafi Bilgi Sistemleri Şube Müdürlüğü ve taşra teşkilatının yoğun çalışmaları ile oluşturulmuş güncel verilerdir. Kullanılan verilerin saklandıkları tablolardaki satır sayıları ve veri tipi Çizelge 5.1.'de verilmiştir. Nehir_Link ve Nehir_node tabloları oluşturulan Network Data Model bölümünde detaylı bir şekilde anlatılmıştır.

Çizelge 5.1. Çalışmada kullanılan Veriler ve Özellikleri

Tablo Adı	**Satır Sayısı**	**SGO_GEOMETRY veri tipi**
Havzalar	28 (Bir havza alt katmanları olması sebebiyle)	Polygon
AGI	2495	Point
Akarsu	23 797 (Nehir, Dere, Çay…)- Bir Akarsu yaklaşık 140 Line Segmentten oluşmaktadır.	LineString
İller	81	Polygon
Göller	132	Polygon
Baraj	1289	Polygon
Nehirler	1902	Linestring
Nehir_Link	118	Linestring
Nehir_Node	121	Point

Coğrafi analizlerin hazırlanılmasında kullanılan temel coğrafi katmanlar da aşağıda kısaca tanımlanmaktadır.

Havzalar: Havzalar, toplanan tüm suyun biriktirildiği ve toplanıp kendinden daha büyük bir su kaynağına doğru akmak itibariyle suyun boşaltıldığı alanlar olarak tanımlanmakta olup, ülkemiz üzerinde 26 temel havza tanımlanmış bulunmaktadır. Havza alanları aşağıda Şekil 5.1.'de "Google Earth" üzerinde gösterilmektedir.



Şekil 5.1. 26 Adet Havza'nın Google Earth üzerinde gösterimi

Akım Gözlem İstasyonları (AGİ): Akarsu ve göllerde, suyun debisinin, seviyesinin ve bazı hidrolojik verilerin ölçüldüğü istasyonlar olup bazı AGİ'lerde suyun kalitesinin saptanması için kalite ölçümleri de yapılmaktadır. Halihazırda, 2000'den fazla AGİ ve bunlara ait öznitelik verisi veritabanında bulunmaktadır.

Akarsu: Yerüstünde veya yeraltında akan bütün sulara verilen genel ad olup bu çalışma kapsamında yaklaşık 24 000 akarsu verisi içerisinde "Oracle" tarafından sağlanan "Network Data Model" ile seçilen bir akarsu verisi yönlü çizge şeklinde ifade edilerek kullanılmıştır.

İller: Ülkemiz, halihazırda 81 ile bölünmüş olup veritabanımıza bu illerin coğrafi sınırları katman olarak eklenmiştir.

Göller: Bu uygulama kapsamında Ülkemiz üzerinde 132 adet göl verisi kullanılmıştır.

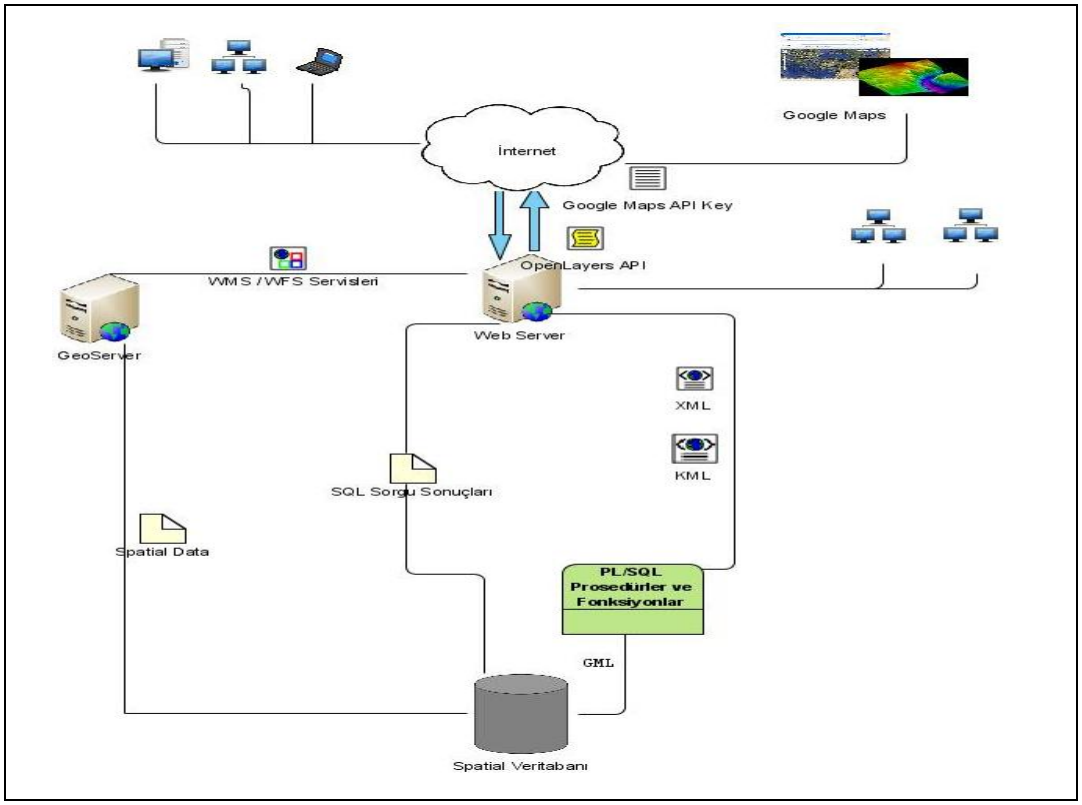
Barajlar: İnşaatı devam eden ve tamamlanmış barajlar veritabanında saklanmaktadır.

Nehirler: Nehirler tablosunda sadece Nehir tipindeki akarsular saklanmıştır.

Yukarıda belirtilen veriler üzerinde hem veritabanı yapısı hem de uygulama tarafı üzerinde geliştirilen çalışmalar birleştirilmek sureti ile mekansal sorgular yapacak araçlar geliştirilmiş olmakla birlikte bu verilere test amaçlı kayıtlar eklenerek de sorgu performansı değerlendirilmiştir. Yapılan performans test sonuçları dikkate alınarak iyileştirme çalışmaları yapılmıştır.

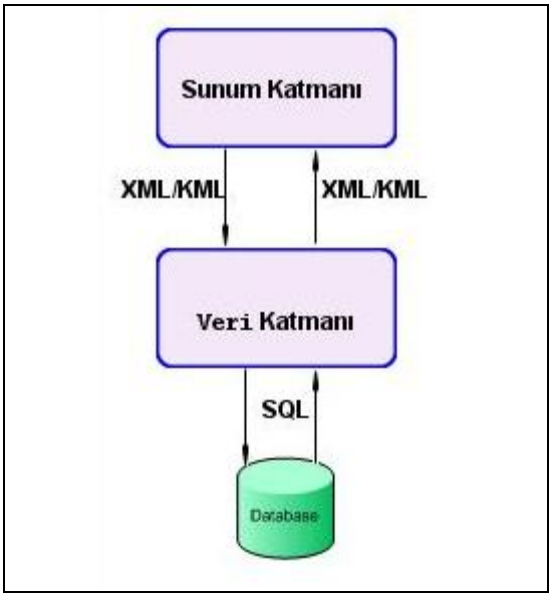
5.2. Kullanılan Sistemin Genel Yapısı

Bu tez kapsamında oracle veritabanı kullanılarak mekansal veriler için sağladığı veri tipi, fonksiyon ve prosedülerden yararlanılmış olup, Microsoft ASP.NET teknolojisi ile bir uygulama geliştirilmiş ve coğrafi verilerin yayımlanmasına izin veren Geoserver kurularak bazı altlıkların bu sunucu üzerinden alınması sağlanmıştır. Web tabanlı bu uygulamada mekansal verinin görüntülenmesi için tamamen javascript ile geliştirilmiş, açık kaynak kodlu "Openlayers API"si kullanılmıştır. Söz konusu "JavaScript API"ye internet üzerinden serbest olarak ulaşılabilmekte ve temin edilebilmektedir. Ayrıca, "Googlemaps API"si kullanılarak Googlemaps altlıkları da uygulama içinde kullanılmıştır. Şekil 5.2.'de uygulamanın genel yapısı görülmektedir.



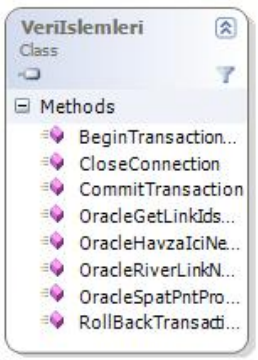
Şekil 5.2. Geliştirilen uygulamanın genel yapısı

Tez kapsamında hazırlanan web uygulaması "Microsoft .NET Framework", "ASP.NET", "C# dili", "Google API" ve "OpenLayers API" kullanılarak geliştirilmiştir. Uygulama; Veri (Data) ve Sunum (Presentation) katmanlarından oluşmaktadır. Şekil 5.3.'de uygulamanın katman yapısı genel olarak gösterilmektedir.



Şekil 5.3. Katman Yapısı

Veri Katmanı: Bu katman, veritabanı bağlantısının kurulduğu, veritabanı prosedürlerinin ve çeşitli SQL cümlelerinin çalıştırıldığı, kısaca veritabanı işlemlerinin yapıldığı katmandır. Şekil 5.4.'de projede kullanılan Veri katmanında kullanılan sınıfların (class) bazı metodları görülmektedir.



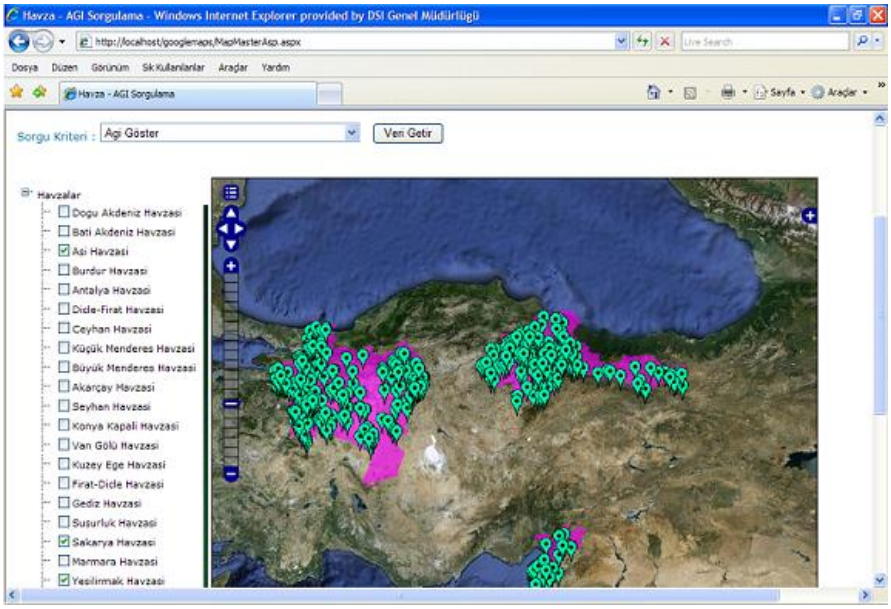
Şekil 5.4. Veri Katmanında Kullanılan Sınıflar

Veri katmanında kullanılan bir metod Şekil 5.5.'de gösterilmiştir. Bu metodda da görüldüğü gibi "HAVZALAR" prosedürü program tarafından parametre olarak gönderilen Havza numaraları ve Dosya adları ile çalıştırılmaktadır. Bu prosedür sonucunda üretilen XML ya da KML dosyası program üzerinden gösterilmetedir.

```
public int OracleHavzalarProcedureCalistir(string Havza_no,
string DosyaA)
        {
try
            {
 OracleCommand objCmd = new OracleCommand();
 objCmd.Connection = cn;
 OpenConnection();
 objCmd.CommandText = "HAVZALAR";

 objCmd.CommandType = CommandType.StoredProcedure;
objCmd.Parameters.Add("Havza_no", OracleType.VarChar).Value =
Havza_no.ToString();
objCmd.Parameters.Add("DosyaA", OracleType.VarChar).Value =
DosyaA.ToString();
objCmd.ExecuteNonQuery();

            }
            catch (Exception e)
            {
                throw Hata(e, "Hata");
            }
            finally
            {
                CloseConnection();

            }}
```

Şekil 5.5. Program içerisinden Prosedür çağrılması

Sunum Katmanı: Sunum katmanı, hazırlanan verilerin kullanıcılara gösterildiği katmandır. Bu Tez kapsamında geliştirilen ve Şekil 5.6.'da görünen web sayfası sunum katmanına örnek olarak verilebilir. Veritabanı prosedürleri ile oluşturulan XML/KML dosyalarının sunum katmanında gösterilmesi, Openlayers Api'si ve Google maps Api'si kullanılarak javascript kodları ile olmaktadır. Yine yazılım üzerinden seçilen kriterlere göre prosedür yardımı ile oluşturulan Akım Gözlem İstasyonlarının gösterimine ilişkin fonksiyon Ek F'de görülebilir. Bu katmanda Geoserver üzerinden WMS servisi ile altlıklar alınmakta, istenirse google maps harita altlıkları ya da Geoserver altlıkları arasında geçiş yapılabilmektedir.



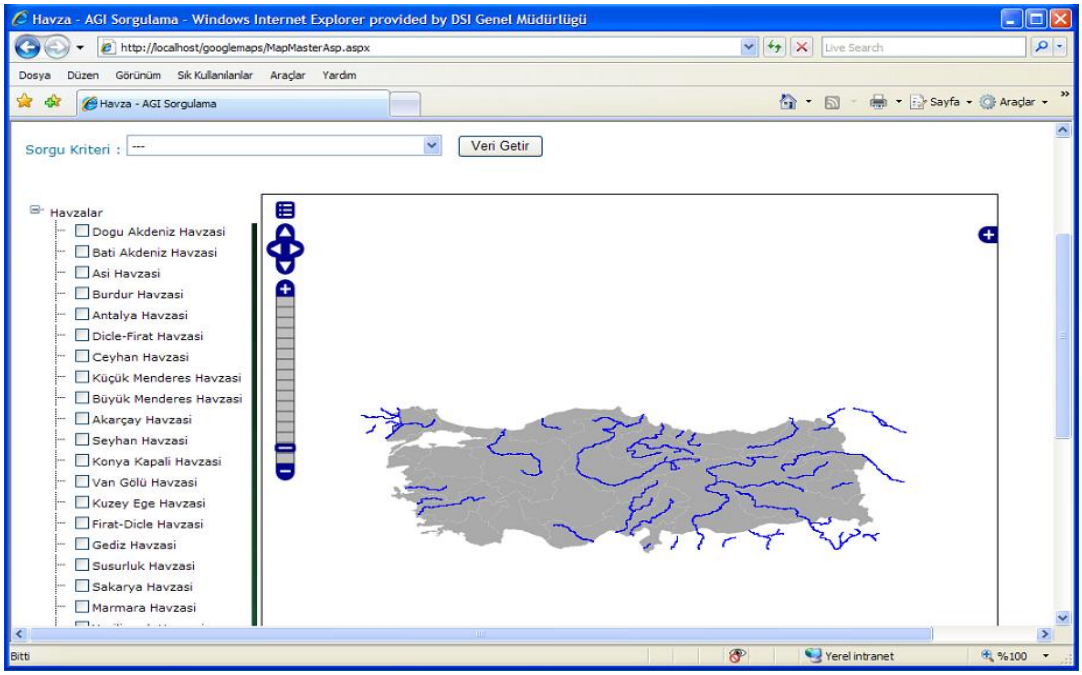
Şekil 5.6. Sunum Katmanından Bir Sayfa Görüntüsü

Geoserver üzerinde tanımlanan katmanlar Javascript Openlayers Api'si kullanılarak uygulamaya eklenmektedir. Şekil 5.7.'de görünen openlayers script sayfaya eklenerek WMS servisi ile havzalar katmanı uygulamada gösterilmektedir.

```
tiled = new OpenLayers.Layer.WMS(
                    "topp:havzalar - Tiled",
"http://localhost:8989/geoserver/wms",
                    {
                        width: '800',
                        layers: 'topp:havzalar',
                        styles: '',
                        srs: 'EPSG:4326',
                        height: '300',
                        format: format,
                        tiled: 'true',
                        tilesOrigin : "25.0977295,35.4999238"
                    },
                    {buffer: 0}
                );
map.addLayer(tiled);
```

Şekil 5.7. WMS servisi ile havzalar'ın Web sayfası üzerinde gösterimini sağlayan Javascript kodu

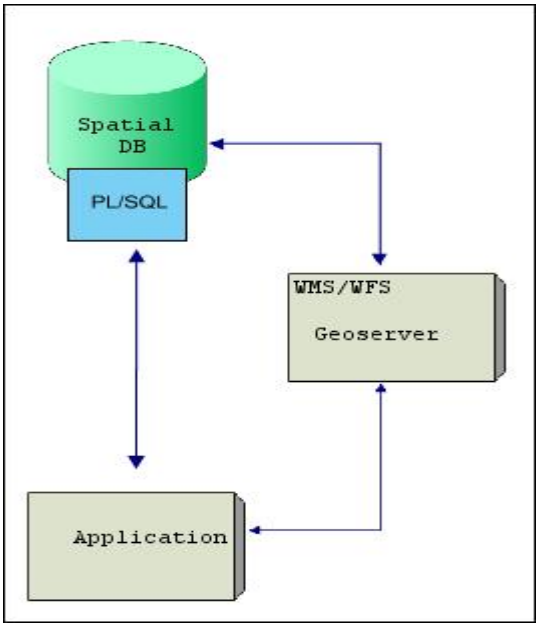
Şekil 5.8.'de Geoserver'dan WMS servisi ile alınan Havzalar ve Nehirler katmanları görülmektedir.



Şekil 5.8. Havzalar ve Nehirler katmanı

5.3. Uygulama Üzerinden Yapılan Sorgulamalar

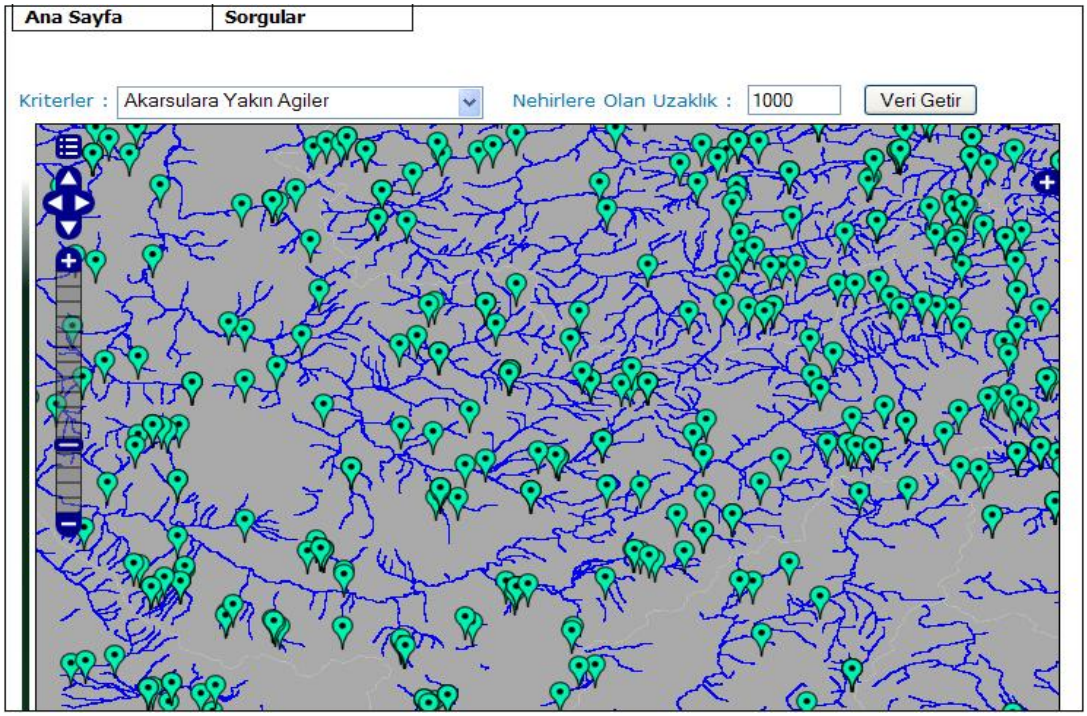
Yukarıdaki bölümlerde anlatılan verilere ait çeşitli sorgulamalar uygulama üzerinden çalıştırılarak çeşitli analizler yapılmaktadır. Uygulamanın genel mimari yapısına baktığımızda, çalıştırılan prosedürler ve SQL sorguları ile Geoserver üzerinden altlıklar alınarak web tarayıcı üzerinde coğrafi veriler gösterilmektedir. Bu yapı Şekil 5.9.'da gösterilmiştir. Çalışma kapsamında tasarlanan web sayfaları ve çalıştırılan sorgular aşağıda belirtilmiştir.



Şekil 5.9. Coğrafi veri altyapısı mimarisi

5.3.1. Akarsulara Yakın AGİ'lerin Bulunması

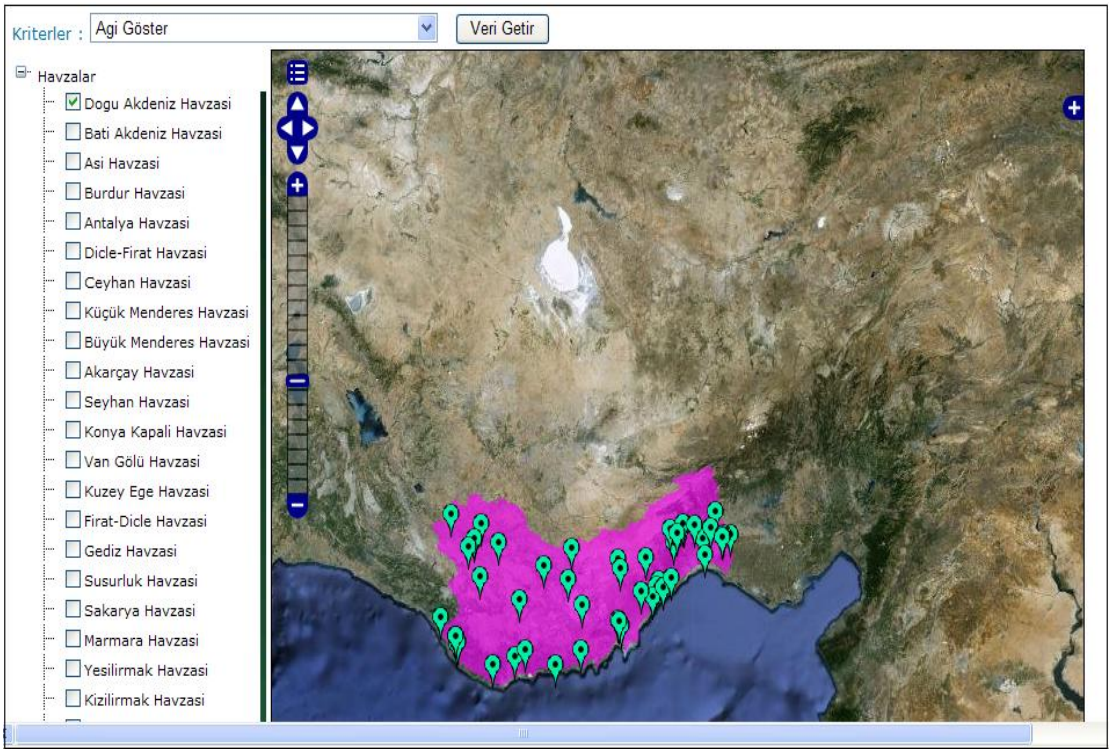
Veritabanında tanımlı tüm akarsulara en yakın Akım Gözlem istasyonlarının bulunması amacıyla bu sayfa tasarlanmıştır. Şekil 5.10.'da da görüldüğü gibi nehirlere yakın AGİ'lerin bulunması için, nehre ne kadar uzaklıktaki Akım Gözlem İstasyonlarının bulunması isteniyorsa bu mesafe prosedüre parametre olarak gönderilmek üzere girilir. Sorgu çalıştırılır ve Akarsulara verilen mesafede bulunan AGİ'ler sayfada gösterilir.



Şekil 5.10 Akarsulara yakın AGİ'lerin bulunması için tasarlanmış web sayfası görünümü

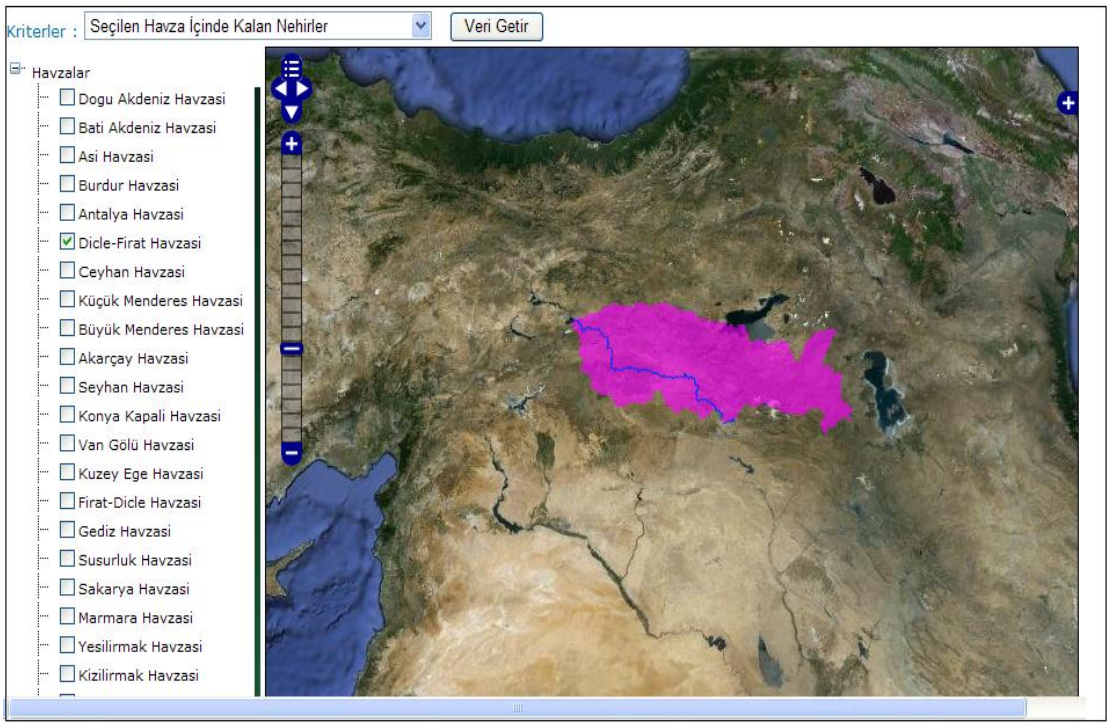
5.3.2. Havzalar, Havza İçinde Kalan AGİ'ler ve Nehirlerin Bulunması

"Havza sınırlarının gösterilmesi", "Havza sınırları içinde kalan AGİ'lerin tespiti" ve "Havza sınırları içerisinde kalan Nehirlerin tespit edilmesi" amacıyla bu sayfa tasarlanmıştır. Sorgu kriterleri seçildikten sonra, ilgili parametreler XML/KML dosyaları üretilmek üzere prosedüre gönderilir. Javascript komutları ile de sayfa üzerinde sonuçlar gösterilir. Şekil 5.11.'de seçilen bir Havza sınırları içinde kalan AGİ'ler gösterilmektedir.



Şekil 5.11. Havza sınırları içinde kalan AGİ'lerin bulunması için tasarlanmış web sayfasından görünüm

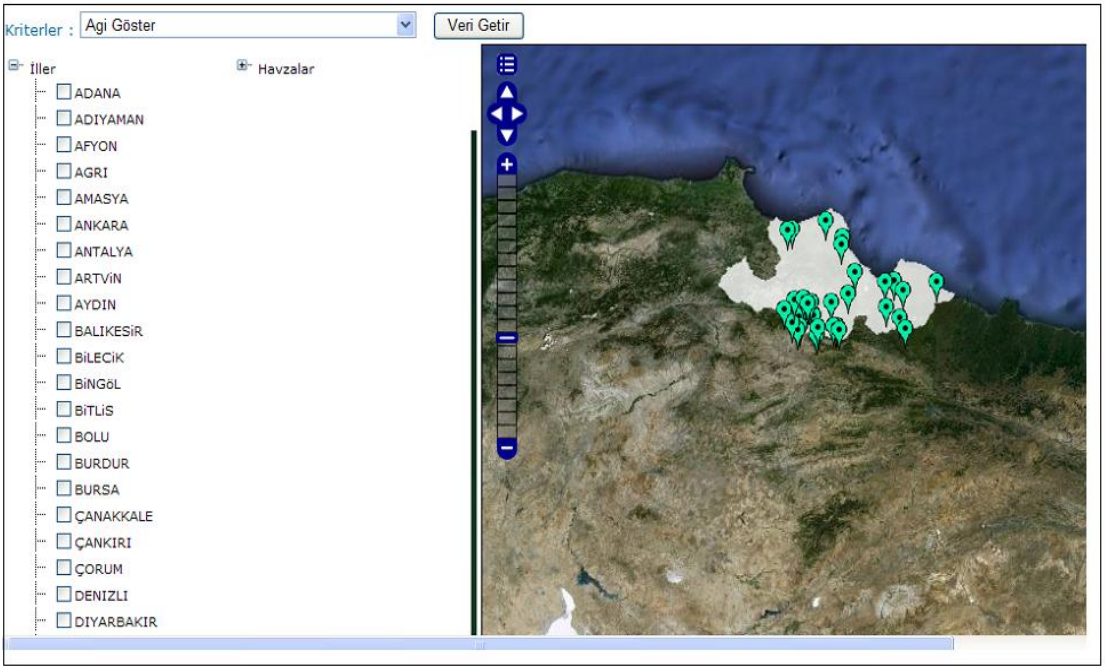
Yine aynı sayfa üzerinden sorgulanabilen "Havza içinde kalan nehir geometrileri" sorgusu sonucu da Şekil 5.12.'de gösterilmektedir.



Şekil 5.12. Havza sınırları içinde kalan Nehirlerin bulunması için tasarlanmış web sayfası görünümü

5.3.3. İller içinde kalan AGİ'ler ve Havzalar İçinde Kalan İller

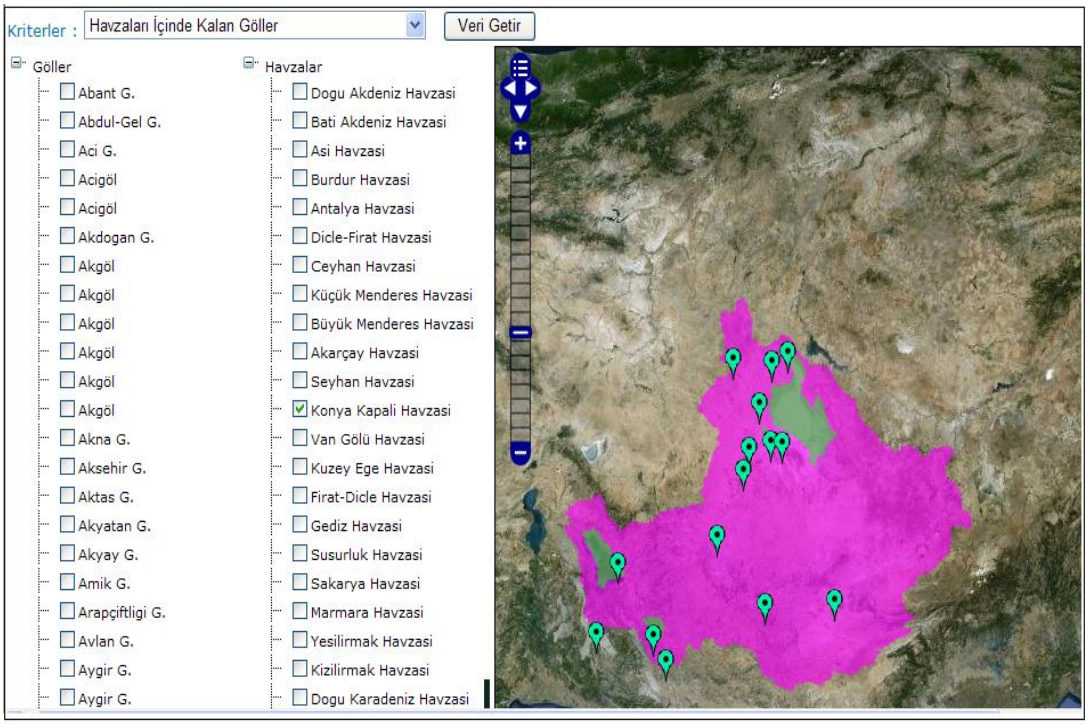
"Seçilen İl ya da İllerin gösterilmesi", "Seçilen İl ya da İl sınırları içinde kalan AGİ'lerin tespit edilmesi", "Havza İçinde kalan İllerin gösterilmesi" amacıyla bir sayfa tasarlanmıştır. İl ve Havzalar seçilerek sorgular çalıştırılır. Sorgu kriterleri ilgili prosedürlere parametre olarak gönderilerek XML/KML dosyaları elde edilir. Şekil 5.13.'de seçilen Samsun ili sınırları içerisinde kalan AGİ'ler görülmektedir.



Şekil 5.13. İl sınırları içinde kalan AGİ'lerin bulunması için tasarlanmış web sayfası görüntüsü

5.3.4. Göl içinde kalan AGİ'ler ve Havzalar İçinde Kalan Göller

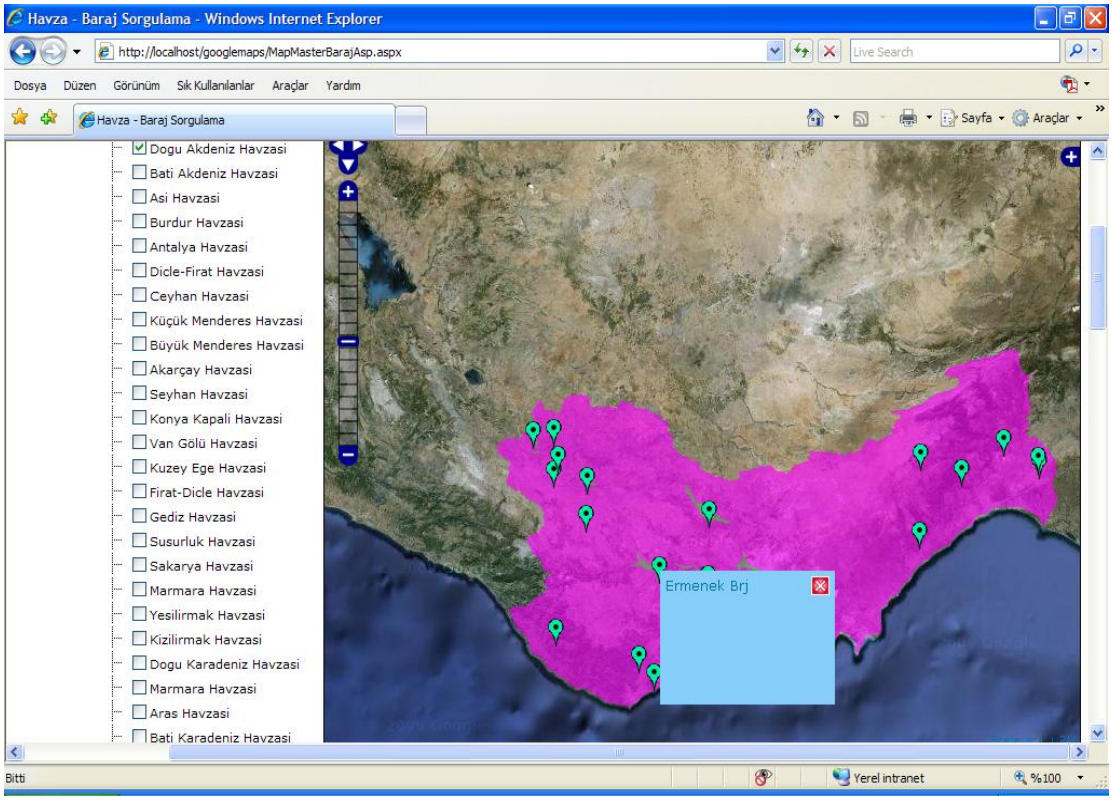
"Göller içerisinde kalan Akım Gözlem İstasyonlarının bulunması" ve "Havzalar içerisinde kalan göllerin tespit edilmesi" amacıyla bu sayfa tasarlanmıştır. Seçilen bir havza içerisinde kalan göller ve göller içerisinde bulunan Akım gözlem istasyonları Şekil 5.14.'de gösterilmiştir.



Şekil 5.14. Havza sınırları içinde kalan Göl'lerin bulunması için tasarlanmış web sayfası görüntüsü

5.3.5. Barajlar içinde kalan AGİ'ler ve Havzalar İçinde Kalan Barajlar

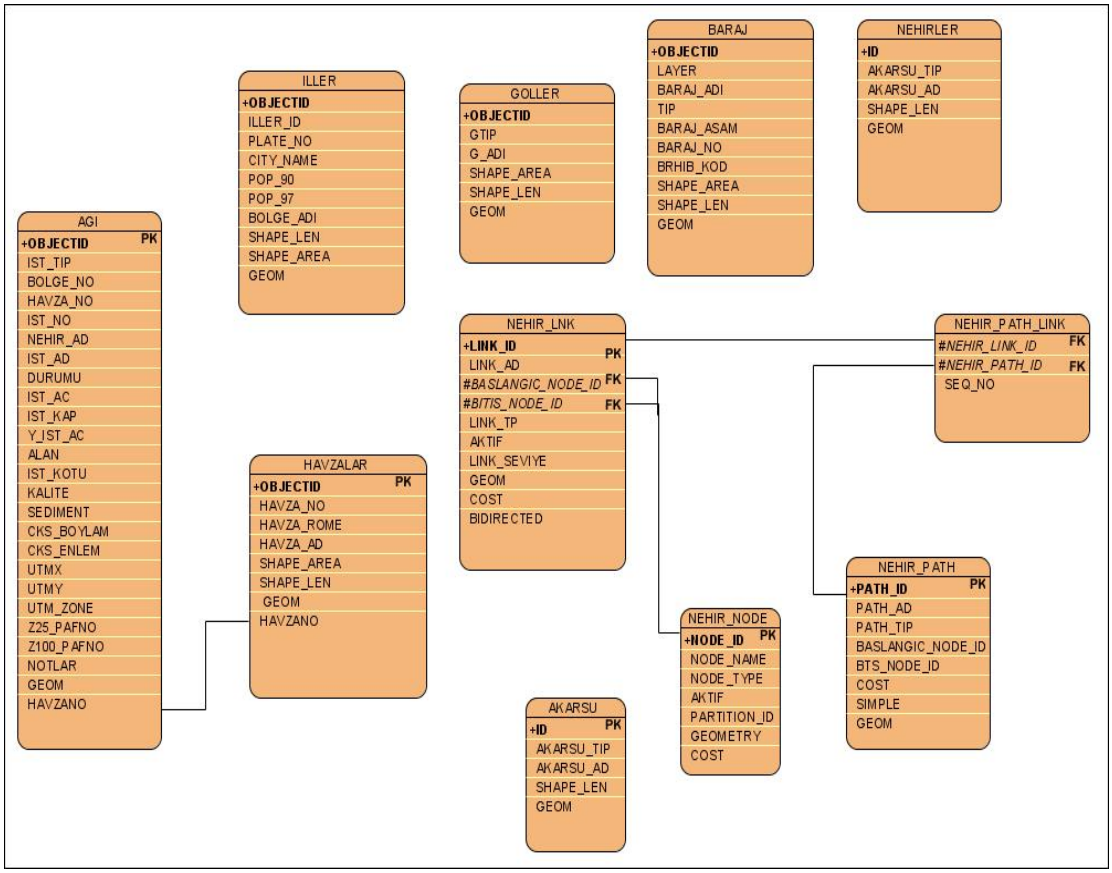
"Barajlar içerisinde kalan Akım Gözlem İstasyonlarının bulunması" ve "Havzalar içerisinde kalan barajların tespit edilmesi" amacıyla bu sayfa tasarlanmıştır. Seçilen bir havza içerisinde kalan barajlar ve barajlar içerisinde bulunan Akım gözlem istasyonları Şekil 5.15.'de gösterilmiştir.



Şekil 5.15. Havza sınırları içinde kalan Baraj'ların bulunması için tasarlanmış web sayfası görünümü

5.4. Veritabanı Şeması

Tez kapsamında mekansal verilerin depolanması, sorgulanması ve çeşitli analizlerin yapılması maksadıyla Oracle veritabanı seçilmiş ve 10g Release 2 (10.2) versiyonu kullanılmıştır. Kullanılan veritabanı tabloları Şekil 5.16.'da gösterilmiştir.

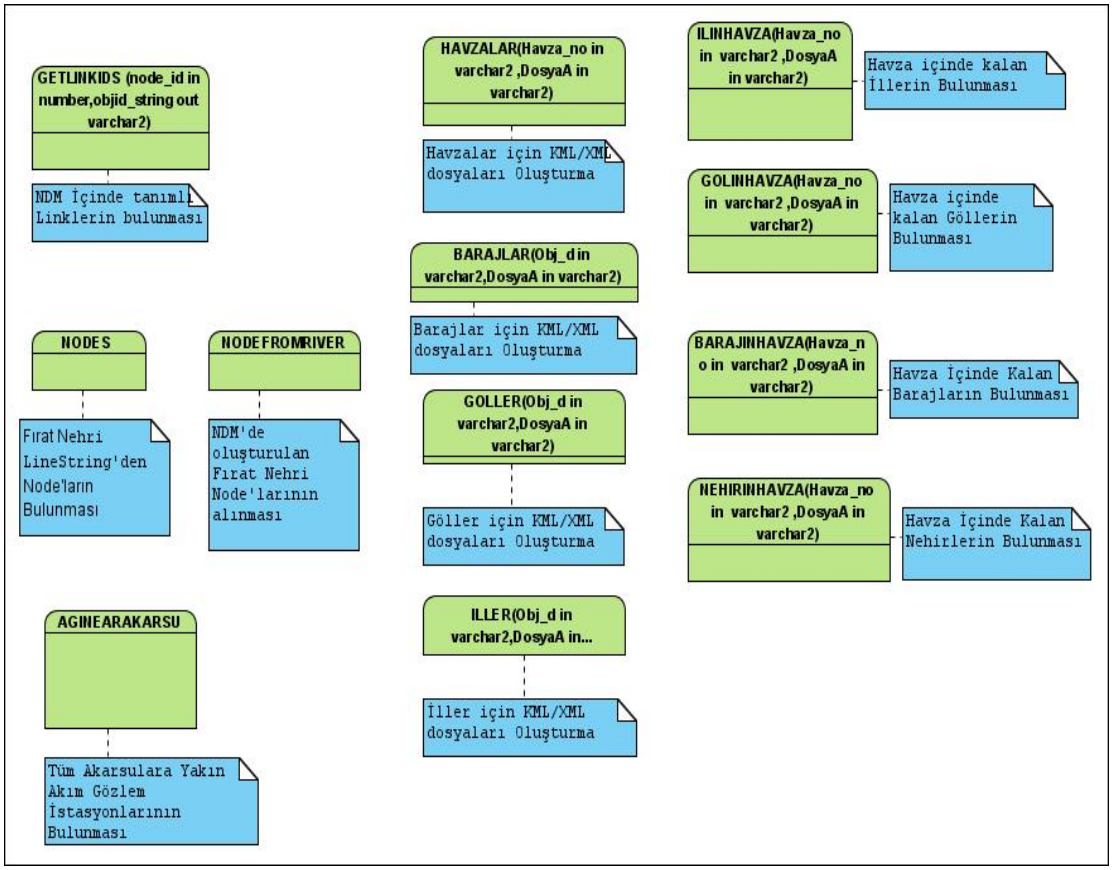


Şekil 5.16. Veritabanı Şeması

Veritabanında kullanılan tabloları incelediğimizde;

Polygon(alan) geometri tipinde Havza, İl, Göl ve Baraj verileri, Point(nokta) geometri veri tipinde Akım Gözlem İstasyonları-AGİ verileri ve Linestring(çizgi) geometri tipinde Akarsu verileri kullanıldığından daha önceki bölümlerde bahsedilmişti. Ayrıca Oracle tarafından sunulan Network Data Model, çalışma kapsamında kullanılmış ve Fırat nehri örnek çalışma olarak seçilmiştir.

Veritabanında birçok analiz için prosedürler kullanılmıştır. Bu prosedürler yardımı ile Havza, İl, Göl, Akarsu ve Akım Gözlem İstasyonlarına ait analizler yapılarak, bu geometriler için XML ve KML dosyaları oluşturulmuştur. Kullanılan Prosedürlerin listesi Şekil 5.17.'de görülmektedir.



Şekil 5.17. Çalışmada kullanılan Stored Prosedürler

Çalıştırılan bazı prosedürler aşağıda detaylı olarak incelenmiştir.

Havza verilerinin xml veya kml formatına dönüştürülmesi:

Dönüşüm işlemi stored prosedür yardımıyla yapılmış, sonuçlar xml veya kml formatında bir dosyaya yazdırılmıştır. Bu prosedür içinde kullanılan SQL cümlesi Şekil 5.18.'de gösterilmektedir. Buna göre, uygulama üzerinde prosedüre, havzalar tablosuna ait seçilen havza numaraları "Havza_no" parametresi ile, oluşturulacak dosyanın adı da "DosyaA" parametrisi ile gönderilerek prosedür çalıştırılır ve oluşan dosyanın web tarayıcı üzerinden DOM komutları ile ulaşılarak gösterilmesi sağlanır.

Oracle SDO_GEOMETRY veri tipindeki veriler ancak GML formatında alınabilmekte bu nedenle de xml veya kml formatında veri formatına dönüşüm işlemlerinin yapılması gerekmektedir. Oracle 11g ile bu dönüşüm işlemleri SDO_UTIL paketine eklenmiş olsa da, bu tez Oracle 10g ile geliştirilmiş olduğundan

bu dönüşüm yazılmış olan prosedürlerle yapılmıştır. Prosedürde kullanılan SDO_UTIL.TO_GMLGEOMETRY fonksiyonu ve kullanımı şekil 5.18.'de görülmektedir.

```
select u.HAVZANO HavzaNo, replace( replace(replace(replace(replace(
sdo_util.to_gmlgeometry(     SDO_CS.TRANSFORM(u.geom,     m.diminfo,
8307)),'gml:',''),'</coordinates></coordinates>'),''),'srsName="SDO:
8307"  xmlns:gml="http://www.opengis.net/gml"',''),'  decimal="."
cs="," ts=" "','') gml from HAVZALAR u, user_sdo_geom_metadata m
WHERE m.table_name = 'HAVZALAR' AND m.column_name = 'GEOM' AND
u.havzano =3;
```

Şekil 5.18.GML veri yapısından XML yapısına dönüşümde kullanılan SQL cümlesi

Fırat Nehrine ait node tablosunun oluşturulması:

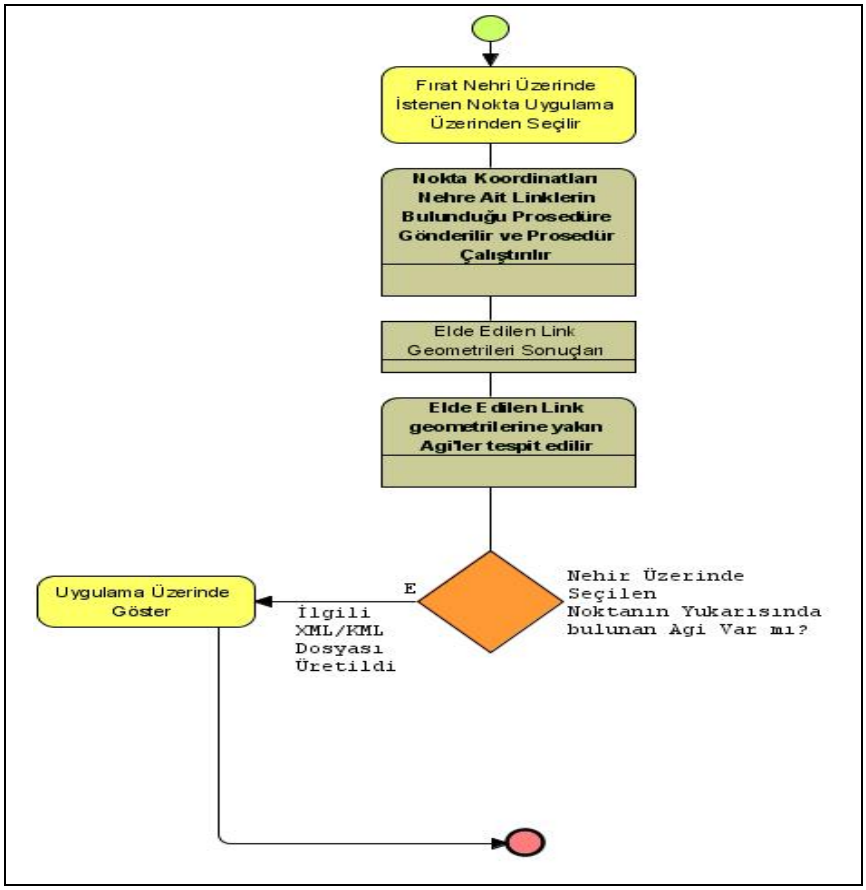
Network Data Model oluşturulurken Fırat nehrine ait LineString tipindeki geometriler bir prosedür yardımıyla NDM'ye ait "node" tablosu için başlangıç ve bitiş noktaları alınarak oluşturulmuştur. Node tablosu çeşitli yollarla oluşturulabilmekle beraber, Şekil 5.19.'da node tablosunu oluşturmak için prosedür içinde kullanılan SQL cümlesi görülmektedir.

```
Select * from table( select sdo_util.getvertices( GEOM ) from
FIRAT    where    objectid=e.objid    and    rownum    <    2    )
where id in ( 1, (select sdo_util.GETNUMVERTICES( geom ) from
FIRAT where objectid=e.objid and rownum < 2 ) )  order by id;
```

Şekil 5.19. NDM node tablosunun oluşturulmasında kullanılan SQL cümlesi

Nehir Üzerinde Seçilen node'un yukarısında kalan AGİ'ların bulunması:

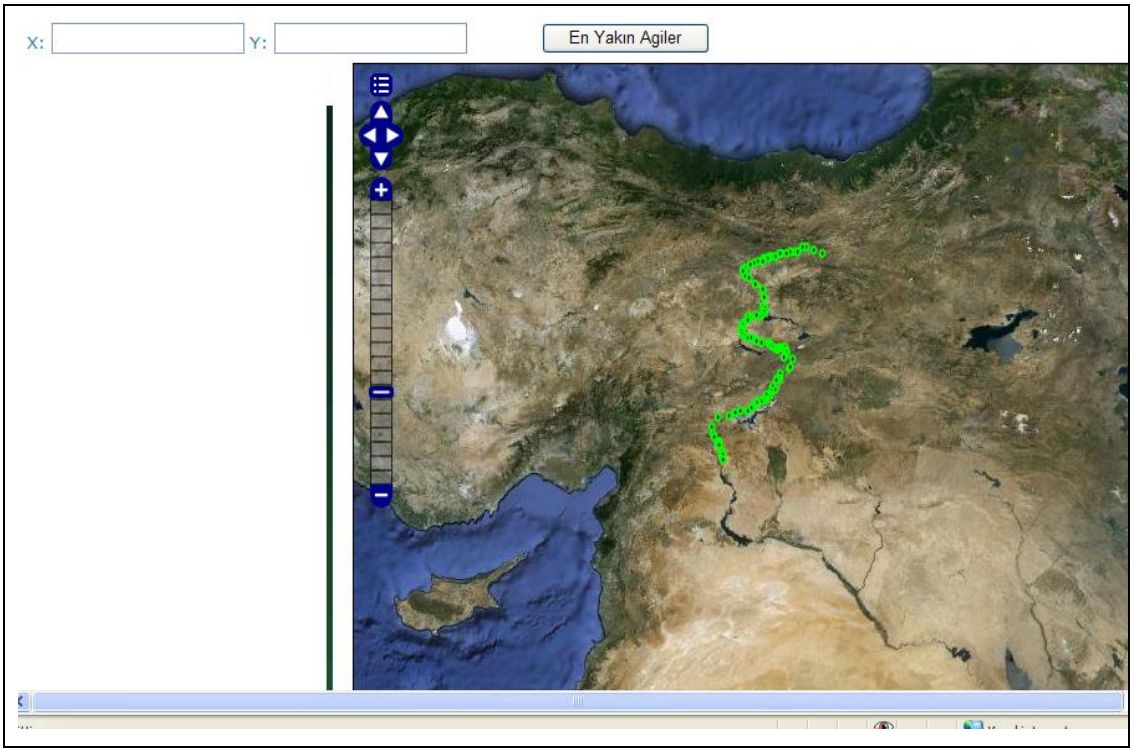
Network Data Model ve oluşturulmasında kullanılan prosedürler hakkında detaylı bilgi altıncı bölümde verilmiştir. Ancak oluşturulan model'in uygulama üzerinden nasıl çalıştırıldığını gösteren akış şeması Şekil 5.20.'de gösterilmektedir.



Şekil 5.20. NDM oluşturulmasını gösteren akış şeması

Bu şemada da görüldüğü üzere;

1. Öncelikle uygulama üzerinden Şekil 5.21.'de görüldüğü gibi nehir üzerinde bulunan node'lar üzerinden seçim yapılır.



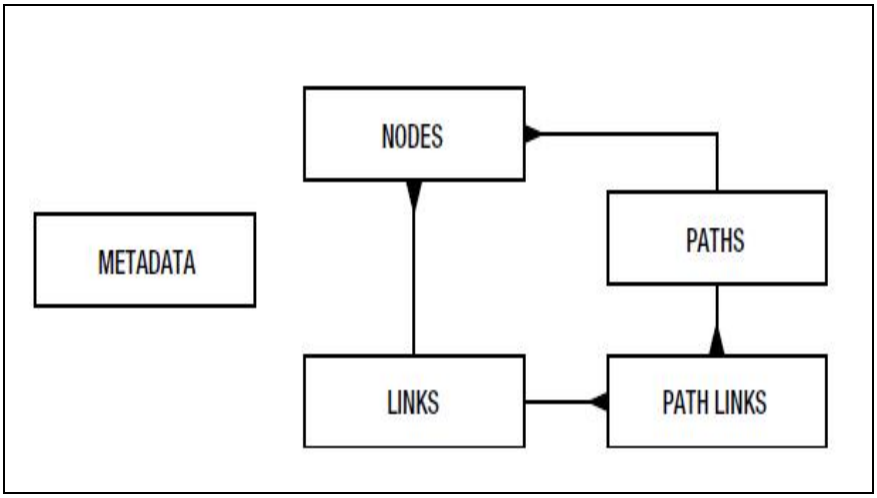
Şekil 5.21. NDM node tablosu verilerinin Google Maps üzerinde gösterilmesi

2. Seçilen noktaya ait koordinatlar prosedürlere gönderilerek, öncelikle bu nokta yukarısında kalan nehir geometrisi tespit edilir. Ek G'de bu prosedür görülebilmektedir.
3. NDM'nin Link tablosundan tespit edilen Linestring nehir geometri Id'leri, diğer prosedüre parametre olarak alınarak en yakın Akım Gözlem İstasyonları bulunur.

5.5. Network Data Model (NDM)

Mekânsal operatör ve fonksiyonlarla birçok analizi yapabiliyor olsak da, tezde de kullanılan bir nehrin aşağısında ve yukarısında kalan AGİ noktalarının bulunması için Oracle Network Data Modeli çok etkin bir yoldur.

Oracle Network Data Modeli bir dizi tablo ile oluşturulmaktadır. Bu Data Modeli Tabloları Şekil 5.22.'de gösterilmektedir. NDM tablolarını manuel olarak oluşturmak da mümkün olabilmektedir. Paths ve Path Links Tabloları opsiyonel'dir.



Şekil 5.22. Network Data Model tablo yapısı

Bu tez kapsamında Fırat Nehri üzerinde yapılacak analizler için manuel Network oluşturulmuştur. Network data modeli oluşturma çalışmasının aşamaları aşağıdaki şekildedir:

1. Network Data Model Oluşturmak için ilk olarak Fırat Nehrine ait linestring veri tipinde tanımlı veriler, node tablosunu oluşturmak amacıyla ilk ve son vertexleri bir prosedür yazılarak Şekil 5.23.'de görünen SQL cümlesi kullanılarak node tablosuna eklenir.

```
Select * from table( select sdo_util.getvertices( GEOM ) from
FIRAT where id=3324 and rownum < 2 ) where id in ( 1, (select
sdo_util.GETNUMVERTICES( geom ) from FIRAT where id=3324 and
rownum < 2 ) )  order by id ;
```

Şekil 5.23. NDM node tablosu oluşturulmasında kullanılan SQL cümlesi

Node tablosu, Link tablosu, path tablosu ve path_link tablolarının manuel olarak yaratılması Ek D'de gösterilmiştir. Link tablosu Fırat Nehri için tanımlı linestring tipinde nehir geometri objeleri, başlangıç ve bitiş node'larına da başlangıç ve bitiş vertexleri girilerek yaratılmıştır.

Network oluşturulurken SDO_NET paketi altında çalıştırılan altprogramlardan SDO_NET.CREATE_SDO_NETWORK prosedürü yardımı ile Network

oluşturulmuştur. Bu prosedürün yapısı ve kullanımına örnek Şekil 5.24.'de görülmektedir.

```
Sql> SDO_NET.CREATE_SDO_NETWORK(
     network IN VARCHAR2,
     no_of_hierarchy_levels IN NUMBER,
     is_directed IN BOOLEAN,
     node_table_name IN VARCHAR2,
     node_geom_column IN VARCHAR2,
     node_cost_column IN VARCHAR2,
     link_table_name IN VARCHAR2,
     link_geom_column IN VARCHAR2,
     link_cost_column IN VARCHAR2,
     path_table_name IN VARCHAR2,
     path_geom_column IN VARCHAR2,
     path_link_table_name IN VARCHAR2);


sql> EXEC SDO_NET.CREATE_SDO_NETWORK(
'RIVER',1,TRUE,'RIVER_NODE','GEOMETRY','COST','RIVER_LINK','GEOMETRY
','COST','RIVER_PATH','GEOMETRY','RIVER_PATH_LINK');
```

Şekil 5.24. Network Data Model oluşturulması

Tablolar yaratıldıktan sonra oluşturulan her Network Data Model için USER_SDO_NETWORK_METADATA view'ına network ile ilgili bilgileri eklemek gerekmektedir. Ayrıca tablolar üzerinde mekansal indeks oluşturmadan önce USER_SDO_GEOM_METADATA tablosuna tablolar ile ilgili gerekli bilgilerin Şekil 5.25.'de görüldüğü gibi eklenmesi gerekmektedir.

```
INSERT INTO USER_SDO_GEOM_METADATA
  VALUES (
  'RIVER_NODE',
  'GEOMETRY',
  SDO_DIM_ARRAY(   -- 20X20 grid
  SDO_DIM_ELEMENT('X', 0, 20, 0.005),
  SDO_DIM_ELEMENT('Y', 0, 20, 0.005)
     ),
  8307   -- SRID (spatial reference system, also called coordinate
system)
);
INSERT INTO USER_SDO_GEOM_METADATA
  VALUES (
  'RIVER_PATH',
  'GEOMETRY',
  SDO_DIM_ARRAY(   -- 20X20 grid
  SDO_DIM_ELEMENT('X', 0, 20, 0.005),
  SDO_DIM_ELEMENT('Y', 0, 20, 0.005)
     ),
  8307   -- SRID (spatial reference system, also called coordinate
system)
);
CREATE INDEX river_node_idx ON river_node(geometry)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX river_link_idx ON river_link(geometry)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

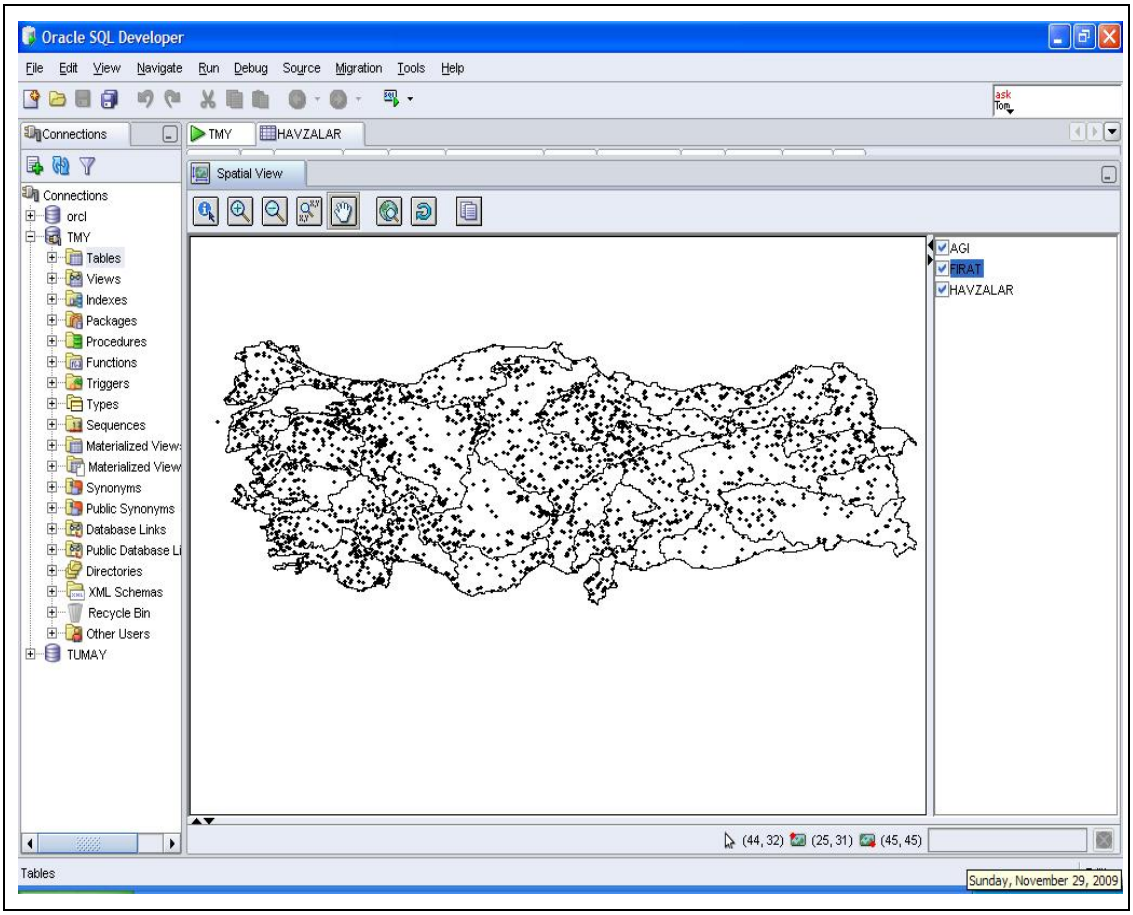
Şekil 5.25. USER_SDO_GEOM_METADATA'ya gerekli bilgilerin eklenmesi

Bu tez kapsamında oluşturulan Network Data Model yardımı ile uygulama üzerinden Fırat nehri üzerinde bir nokta seçilerek, nehrin aşağısında ve yukarısında bulunan en yakın AGİ noktaları yazılan PL/SQL prosedürü ile bulunmuştur. Bu prosedürde SDO_NET_MEM paketi kullanılmıştır.

5.6. Oracle SQL Developer-GeoRaptor

Oracle SQL Developer ücretsiz ve Oracle tarafından desteklenen veritabanı geliştirme aracıdır. SQL Developer ile veritabanı nesnelerine ulaşılabilmekte, SQL cümleleri ve SQL script'leri çalıştırılabilmekte, PL/SQL scriptleri derlenebilmektedir. Tez çalışmaları sırasında kullanılan bir araç olmakla birlikte SQL Developer için ek Geographic Information System-GIS eklentisi olan Georaptor, sağladığı ek GIS fonksiyonları ile tez çalışmaları sırasında yoğun bir şekilde kullanılmıştır. Özellikle Network Data Model oluştururken linestring tipindeki nehir verilerinin başlangış ve bitiş koordinatlarının doğrulanmasında

yardımcı bir araç olmuştur. Şekil 5.26.'da, Oracle SQL Developer içinde Georaptor eklentisinin kullanımı görülmektedir.



Şekil 5.26. Oracle SQL Developer Georapter eklentisi kullanılarak Havza ve AGİ'lerin gösterimi

BÖLÜM 6

VERİTABANI PERFORMANS TESTLERİ

Bu çalışma kapsamında gerçekleştirilen uygulama, 5. bölümde detaylı bir şekilde anlatıldı. Bu bölümde ise veritabanında saklanan coğrafi veriler üzerinde farklı indeksleme mekanizmaları ve tuning araçları kullanılarak yapılan performans testleri ve sonuçları üzerinde durulacaktır.

6.1. R-tree index kullanılarak çalıştırılan SQL'ler ve Test Sonuçları

Veritabanı testleri yapılırken yaygın olarak kullanılan operatörler üzerinde çalışılmıştır. Veriler üzerinde, bu operatörler kullanıldığında nasıl performans değerleri alındığı explain plan, awr raporları, OEM gibi yardımcı araçlarla gözlemlenmiştir. Çalışmada kullanılan SQL cümleleri aşağıda detaylı olarak belirtilmiştir.

6.1.1. Belirli Bir mesafe içinde kalan geometrilerin tespit edilmesi

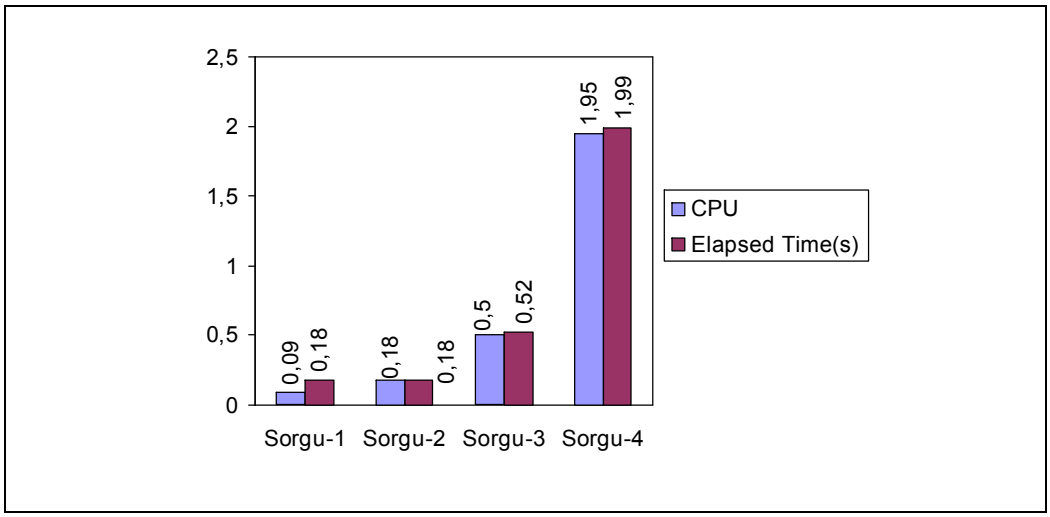
Bu sorgulama için SDO_WITHIN_DISTANCE operatörü kullanılmış ve uygulamaya entegre edilebilmesi için en etkin SQL cümlesinin bulunması amaçlanmıştır. Çizelge 6.1.'de seçilen bir geometriye belirli bir mesafede bulunan geometrilerin tespit edilmesi amacıyla kullanılan birkaç SQL cümlesi görülmektedir.

Çizelge 6.1. Belirli bir mesafe içinde kalan geometrilerin tesbiti için kullanılan SQL'ler

Sorgu No.	Test edilen SQL cümlesi	Açıklama
1.	`SELECT a.objectid FROM river_node rn, agi_ba WHERE rn.node_id=173 AND SDO_WITHIN_DISTANCE(a.geom, rn.geometry, 'querytype=FILTER DISTANCE=30 UNIT=KM' )='TRUE' ORDER BY a.OBJECTID;`	Fırat nehri node'ları üzerinde seçilen bir node'a 30km uzaklıkta bulunan AGİ'lerin bulunması
2.	`SELECT a.objectid FROM river_node rn, agi_ba WHERE rn.node_id=173 AND SDO_WITHIN_DISTANCE (a.geom, rn.geometry, 'DISTANCE=30 UNIT=KM' )='TRUE' ORDER BY a.OBJECTID;`	Fırat nehri node'ları üzerinde seçilen bir node'a 30km uzaklıkta bulunan AGİ'lerin bulunması
3.	`SELECT distinct(i.objectid) FROM akarsu aks, iller i WHERE aks.Akarsu_ad='Çoruh N.' AND SDO_WITHIN_DISTANCE(i.geom, aks.geom, 'querytype=FILTER DISTANCE=100 UNIT=KM')='TRUE' ;`	Çoruh Nehri'ne 100 km. uzaklıkta bulunan illerin tesbiti
4.	`SELECT distinct(i.objectid) FROM nehirler aks,iller i WHERE aks.Akarsu_ad='Çoruh N.' AND SDO_WITHIN_DISTANCE(i.geom, aks.geom, 'DISTANCE=100 UNIT=KM')='TRUE' ;`	Çoruh Nehri'ne 100 km. uzaklıkta bulunan illerin tesbiti

Bu çalışmada kapsamında Fırat Nehri için Oracle Network Data Model oluşturulmuş ve uygulama üzerinden Fırat Nehri node'ları üzerinden seçimler yapılarak nehrin yukarısında ve belirli bir mesafe yakınında olan Akım Gözlem İstasyonları tespit edilmiştir. Bu sorgu için öncelikle Çizelge 6.1.'de görünen 1. ve 2. SQL cümleleri denenmiştir. SDO_WITHIN_DISTANCE operatörü 4. bölümde de anlatıldığı gibi birincil ve ikincil filtre seçimine, sorgu cümleciğinde kullanılan parametreye göre karar vermektedir. Burada 1. SQL'de görünen "querytype" parametresi ile birincil filtre kullanımı sağlanmış ve 2. SQL cümlesinde "querytype" paremetresi set edilmeyerek ikincil filtre kullanımı sağlanmıştır. Bu SQL cümleleri çalıştırıldığında elde edilen trace file sonuçlarına göre bulunan CPU ve Elapsed time değerleri Şekil 6.1'de görülmektedir. Ayrıca 1. SQL cümlesi için üretilmiş olan Explain Plan ve trace file Ek B'de, 2. SQL cümlesi için üretilmiş olan Explain Plan ve trace file sonucu ise Ek C'de görülebilmektedir.

Yapılan sql trace ve explain planlar da incelendiğinde 1. sorgunun %50 oranında daha hızlı çalıştığı görülmüştür. Bu da gösteriyor ki eğer aynı sonucu üretmemizi sağlıyorsa birincil filtre kullanımı daha hızlı sonuç üretmektedir.



Şekil 6.1. SDO_WITHIN_DISTANCE Operatörü kullanılarak çalıştırılan SQL performans sonuçları

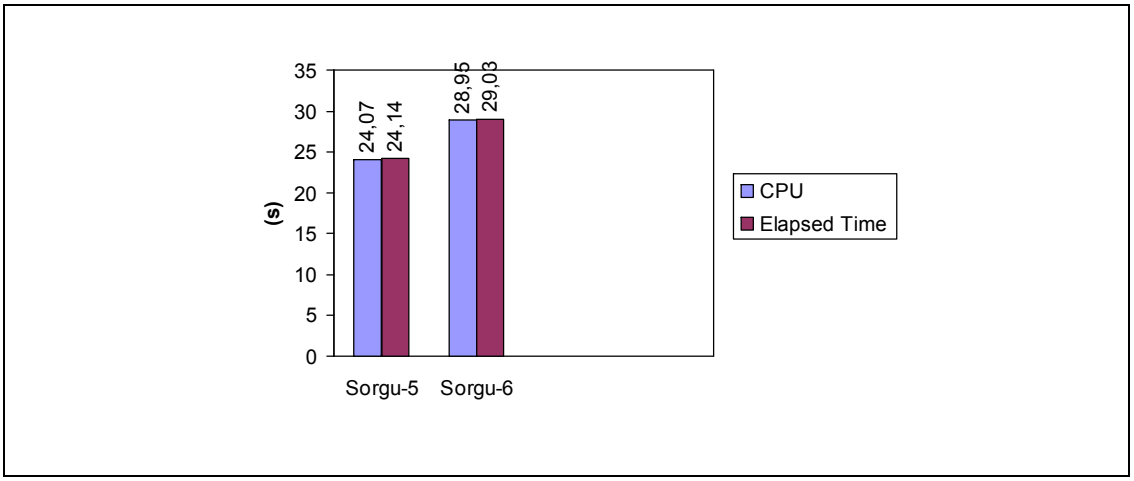
Operatör, ikinci olarak "Nehirler tablosunda seçilen Çoruh Nehrine 10 km uzaklık içinde kalan İllerin saptanması'nda kullanılmıştır. Burada da yine SDO_WITHIN_DISTANCE operatörü kullanılmış, sorgu için querytype parametresi set edilerek(primary filter) ve set edilmeyerek (primary ve secondary filter) performans değerlendirmesi yapılmıştır. Trace file sonucu elde edilen değerler Şekil 6.1'deki grafikte 3 ve 4 numaralı sonuçlar olarak gösterilmiştir. Buna göre yaklaşık %50 oranında Çizelge 6.1.'de 3. SQL'in performansının daha iyi olduğu görülmüştür. 4. SQL kesin sonuçları vermekle birlikte kritik bir durum oluşturmuyorsa yaklaşık sonuçlarının üretildiği 3. SQL'in tercih edilmesi önerilmektedir.

Çizelge 6.2. SDO_WITHIN_DISTANCE "ORDERED"hint'i için kullanılan SQL'ler

Sorgu No	Test edilen SQL cümlesi	Açıklama
5.	`SELECT /*+ ORDERED */ a.objectid FROM akarsu r, agi a WHERE SDO_WITHIN_DISTANCE (a.geom, r.geom, 'DISTANCE=200 UNIT=METER ' )='TRUE';`	Akarsulara 200 metre mesafede bulunan AGİ'lerin tesbiti ordered hint'i kullanımı
6.	`SELECT a.objectid FROM akarsu r, agi a WHERE SDO_WITHIN_DISTANCE (a.geom, r.geom, 'DISTANCE=200 UNIT=METER ' )='TRUE';`	Akarsulara 200 metre mesafede bulunan AGİ'lerin tesbiti

Optimizer tabloların çok büyük olmadığı durumlarda nested-loop join'i tercih etmektedir. Oracle, nested-loop joinde bir tabloyu outer tablo(driving table) olarak almakta ve driving tablonun her satırı için inner tablonun satırlarını fetch etmektedir. Burada inner tablonun from cümleciğinde en sonda gelen tablo olması gerekmektedir. Eğer SQL cümleciğinde birden fazla tablo kullanılmışsa optimizer spatial indeksi kullanmayabilir ve bu durumda "ORDERED" hint'i yardımı ile Optimizer'ı bilgilendirerek, tanımlı indekslerin her zaman kullanımı tetiklemek gerekir. Bu SQL cümlesini de de Çizelge 6.2.'de 5 numaralı sorguda görüldüğü gibi yazabilmektedir. SDO_WITHIN_DISTANCE operatörü, AGI tablosundaki veriler yaklaşık 5 katı kadar artırılarak Çizelge 6.2.'de görünen ifadelerle denenmiş ve sonuçlar Şekil 6.2.'deki grafikte gösterilmiştir. Grafikte de görüldüğü gibi ORDERED hint'i kullanımı performans artışı sağlamaktadır.

Burada grafiklerde dikkat çeken bir nokta da bazı durumlarda cpu zamanının, geçen (elapsed) zamandan daha büyük çıkabilmesidir. I/O işlemleri, diskten okuma gibi işlemler elapsed time içerisinde değerlendirilen süreler olmasına rağmen multi-core ya da multi-processor sistemlerde birden fazla CPU'nun kullanılması sebebi ile cpu kullanım zamanı geçen zamandan(*elapsed time*) fazla çıkabilmektedir[44].



Şekil 6.2. SDO_WITHIN_DISTANCE Operatörü "ORDERED" hint'i kullanılarak çalıştırılan SQL performans sonuçları

6.1.2. Tablolardaki Tüm Geometriler İçin Belirli Bir mesafe içinde kalan geometrilerin tespit edilmesi

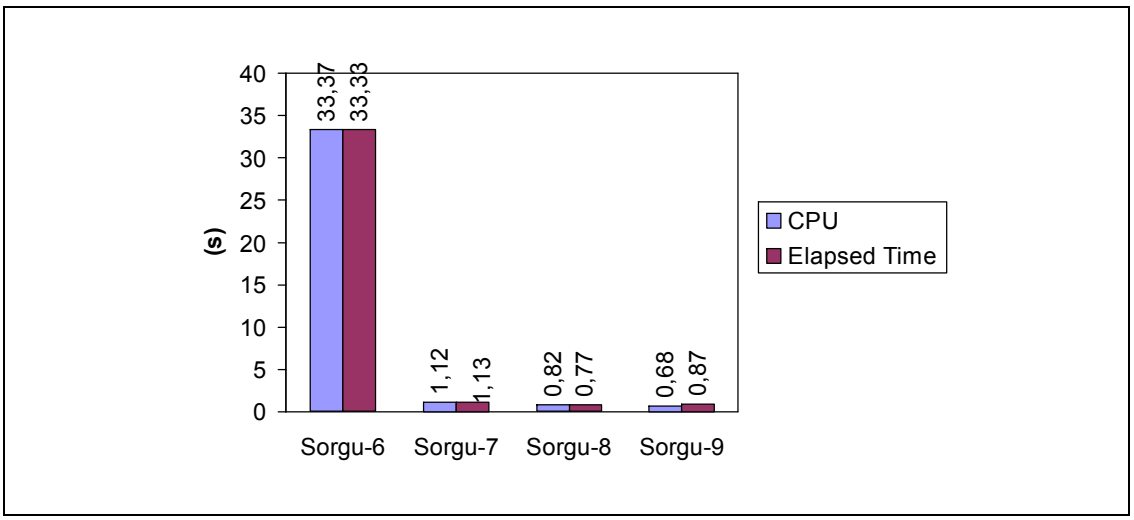
Bu bölümde yapılan çalışma ile eğer bütün tablo elemanları taranacaksa, yani full table scan yapılması gereken bir durum varsa SQL cümlesinin nasıl olması gerektiği belirlenmeye çalışılmıştır. Bu amaçla test edilen SQL cümleleri Çizelge 6.3.'de görülmektedir.

Çizelge 6.3. Tüm Geometriler İçin Belirli Bir mesafe içinde kalan geometrilerin tesbiti için kullanılan SQL'ler

Sorgu No	Test edilen SQL cümlesi	Açıklama
6.	`SELECT a.objectid FROM akarsular aks, agi a WHERE SDO_WITHIN_DISTANCE(a.geom, aks.geom, 'DISTANCE=200 UNIT=METER ' )='TRUE';`	Akarsulara 200 metre mesafede bulunan AGİ'lerin tesbiti
7.	`SELECT a.objectid FROM akarsular aks, agi a , TABLE ( SDO_JOIN ( 'agi', 'geom','akarsular','geom','DISTANCE=200 UNIT=METER') ) jn WHERE jn.rowid1 = a.rowid AND jn.rowid2 =aks. rowid;`	Akarsulara 200 metre mesafede bulunan AGİ'lerin tesbiti
8.	`SELECT a.objectid FROM akarsular aks, agi a ,TABLE ( SDO_JOIN ( 'agi','geom','akarsular','geom', 'DISTANCE=200 UNIT=METER')) jn WHERE aks.AKARSU_AD='Coruh N.' jn.rowid1 = a.rowid jn.rowid2=aks.rowid ;`	Havzalara 200 metre mesafede bulunan AGİ'lerin tesbiti
9.	`SELECT a.objectid FROM akarsular aks, agi a WHERE aks.AKARSU_AD='Coruh N.' and SDO_WITHIN_DISTANCE(a.geom, aks.geom, 'DISTANCE=200 UNIT=METER ' )='TRUE';`	Çoruh Nehrine 200 metre mesafede bulunan AGİ'lerin tesbiti

SDO_JOIN operatörü bir operatörden çok, tablo fonksiyonudur. Ancak kullanımı operatörler gibidir. Sorgu full table scan gerektiriyorsa, SDO_JOIN kullanımı birçok durumda SDO_WITHIN_DISTANCE operatörüne göre daha uygundur. Bu durumun test edilmesi amacıyla Çizelge 6.3.'de görünen sorgular test edilmiştir. "Her bir akarsuya 200 metre uzaklıkta bulunan AGİ noktalarının tespit edilmesi" amacıyla dört adet SQL'den 6 ve 7 numaralı sorgular, eğer where cümleciğinde operatörden başka bir kısıt yoksa SDO_JOIN operatörünün kullanımının daha uygun olduğunu göstermek amacıyla çalıştırılmıştır. Şekil 6-3'de elde edilen CPU ve Elapsed time'lar görülmektedir. Sonuçlar da gösteriyor ki; full

table scan gerektiren durumlarda SDO_JOIN kullanımı SDO_WITHIN_DISTANCE operatörüne göre daha iyi performans göstermektedir.



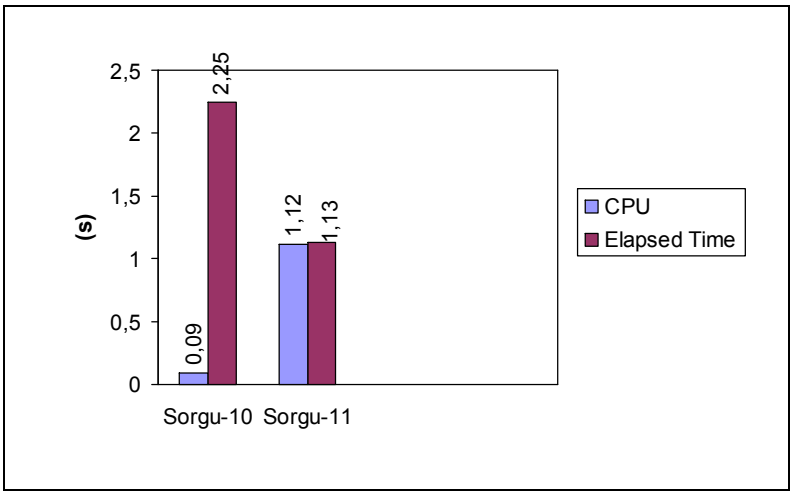
Şekil 6.3. SDO_JOIN Operatörü kullanılarak çalıştırılan SQL performans sonuçları

Çizelge 6.3.'de görünen 8 ve 9 numaralı sorgular çalıştırılarak where cümleciğinde operatör haricinde bir kısıt kulanılmışsa hangi operatörün kullanımının daha uygun olduğu belirlenmek istenmiştir. Burada "Çoruh Nehrine 200 metre uzaklıkta bulunan AGİ noktalarının tespit edilmesi" sorgusu çalıştırılmıştır.

Çizelge 6.4.'de görünen sorgular çalıştırılarak SDO_JOIN operatörü için diğer operatörler için de geçerli olan parametre seçimi ile birincil ve ikincil filtre kullanımlarının test edilmesi çalışması yapılmıştır. Şekil 6-4'deki grafikte de görüldüğü gibi birincil filtrenin kullanıldığı 10 numaralı sorguda cpu zamanının daha az olduğu birincil ve ikincil filtrenin birlikte kullanıldığı 11 numaralı sorgu için ise daha fazla olduğu görülmektedir.

Çizelge 6.4. SDO_JOIN operatörünün “Filter” parametresi ile kullanımı

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
10.	`SELECT a.objectid FROM akarsular aks, agi a , TABLE(SDO_JOIN('akarsular','geom','agi' ,'geom',' mask=FILTER DISTANCE=200 UNIT=METER')) jn WHERE jn.rowid1 = aks.rowid AND jn.rowid2 = a.rowid;`	Akarsulara 200 metre mesafede bulunan Akın Gözlem İstasyonları
11.	`SELECT a.objectid FROM akarsular aks, agi a , TABLE (SDO_JOIN('akarsular','geom','agi', 'geom','DISTANCE=200 UNIT=METER') ) jn WHERE jn.rowid1 = aks.rowid AND jn.rowid2 = a.rowid;`	Akarsulara 200 metre mesafede bulunan Akın Gözlem İstasyonları



Şekil 6.4. SDO_JOIN Operatörü kullanılarak çalıştırılan SQL performans sonuçları

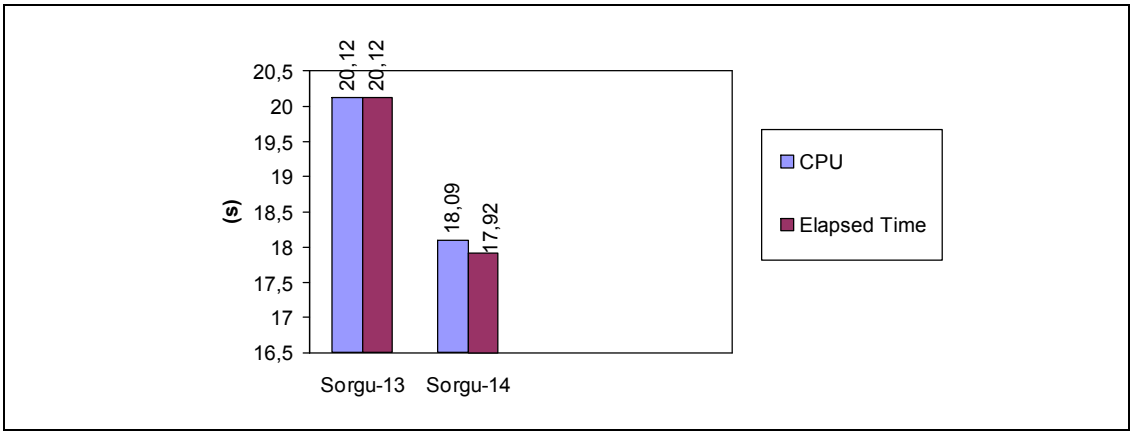
6.1.3. Bir Geometri İçinde Kalan Geometrilerin Tespit Edilmesi

Bu amaçla çeşitli operatör ve fonksiyonlar denenmiştir. Öncelikle " Tüm havzalar içerisinde kalan akarsuların tespit edilmesi" sorgusu için başlangıçta SDO_JOIN operatörü kullanımının uygun olduğu düşünülse de, iki tablonun eleman sayısı arasındaki büyük fark bu analiz için SDO_JOIN operatörünün kullanımını engellemiştir. Tablolardaki eleman sayısı karşılaştırılabilir büyüklükte olması durumunda SDO_JOIN operatörü kullanımı daha hızlı sonuç verse de, bu fark büyük olduğu durumda SDO_JOIN operatörü kullanılan Çizelge 6.5.'de 12 numaralı sorgu çok uzun bir süre sonuç döndürmemiştir. Aynı sorgu SDO_INSIDE ya da SDO_RELATE operatörleri ile daha hızlı sonuç üretmiştir. Sorguların çalıştırılması ile elde edilen sonuçlar Şekil 6.5.'de görülmektedir.

SDO_JOIN operatörü, iki tablo eleman sayısı arasında büyük fark olması durumunda etkin bir çözüm olmamaktadır. Burada Havzalar tablosunun 28, Akarsular tablosunun 23797 eleman sayısına sahip olması SDO_RELATE ve SDO_INSIDE operatörlerinin kullanımını zorunlu kılmıştır.

Çizelge 6.5. Bir Geometri İçinde Kalan Geometrilerin Tespit Edilmesi için denenen SQL'ler

Sorgu No.	Test edilen SQL cümlesi	Açıklama
12.	`SELECT aks.objectid FROM akarsular aks, havzalar hvz, TABLE(SDO_JOIN('akarsular', 'geom', 'havzalar', 'geom','mask=inside')) jn WHERE jn.rowid1 = aks.rowid AND jn.rowid2 = hvz.rowid ;`	Havzalar içerisinde kalan akarsuların tesbiti
13.	`SELECT /*+ ORDERED */ aks.objectid FROM havzalar hvz, akarsular aks WHERE SDO_RELATE(aks.geom, hvz.geom,'mask=inside') = 'TRUE';`	Havzalar içerisinde kalan akarsuların tesbiti
14.	`SELECT /*+ ORDERED */ aks.objectid FROM havzalar hvz, akarsular aks WHERE SDO_INSIDE(aks.geom, hvz.geom) = 'TRUE';`	Havzalar içerisinde kalan akarsuların tesbiti



Şekil 6.5. SDO_JOIN Operatörü kullanılarak çalıştırılan SQL performans sonuçları

6.1.4. Seçilen Bir Geometriye En yakın Geometrilerin Bulunması

Geometriye en yakın noktaların tespit edilmesi amacıyla SDO_NN operatörü kullanılarak "Fırat nehri üzerinde seçilen bir noktaya en yakın Akım Gözlem İstasyonlarının bulunması" sorgusu test edilmiştir. İlk olarak, SDO_NN operatörü ile kullanılan "SDO_BATCH_SIZE" parametresi kullanılarak testler yapılmıştır. Bu amaçla;

Fırat nehrine ait vertexlerin tutulduğu "river_node tablosu üzerinde seçilen bir noktanın 21 no'lu havzada yer alan en yakın 5 AGİ noktasını bulmak" amacıyla Şekil 6.5, 6.6. ve 6.7'de görünen SQL'ler test edilmiştir. Burada amaç "SDO_BATCH_SIZE" parametresi ve "FIRST_ROWS" hint'lerinin kullanımının test edilmesidir. Test sonuçları Şekil 6.9.'da gösterilmiştir.

```
15.SELECT  /*+NO_INDEX*/ a.objectid FROM river_node rn, AGI a
WHERE  rn.node_id=173  and  a.havzano=21  AND  SDO_NN(a.geom,
rn.geometry)='TRUE'  AND  ROWNUM<=5                    ORDER  BY
a.objectid;

call  count     cpu     elapsed    query      current        rows
------------- ------- ----------  ---------- ----------  --------
Parse   1        0.00       0.05         831         0         0
Execute 1        0.00       0.00           0         0         0
Fetch   2        0.40       0.37         131       70523       5

------- ------   ------- ----------    ---------- ---------- -------

total   4        0.40       0.43         962        70523          5
```

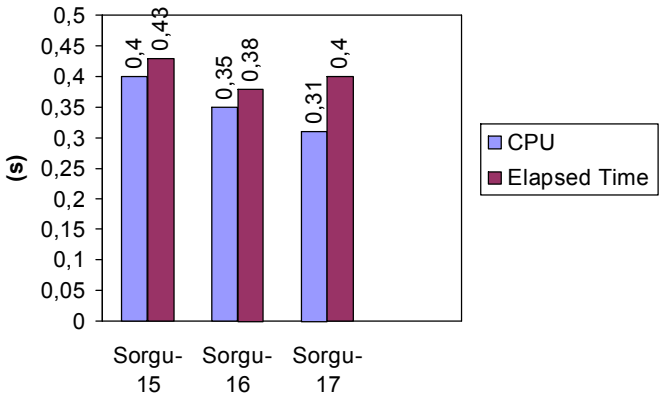
Şekil 6.6. Nehir üzerinde seçilen node'a en yakın 5 noktanın bulunması

```
16.SELECT  a.objectid    FROM  river_node  rn,  AGI  a  WHERE
rn.node_id=173    and    a.havzano=21    AND    SDO_NN(a.geom,
rn.geometry  ,'SDO_BATCH_SIZE=100')='TRUE'     AND  ROWNUM<=5
ORDER BY a.objectid ;

call   count       cpu    elapsed    query    current      rows
----  ------  -------- ----------  ---------- ----------  --------
Parse    1      0.00       0.00      0           0          0
Execute  1      0.00       0.00      0           0          0
Fetch    2      0.35       0.38     200          70523      5
------- ------  -------- ---------- ---------- ----------  -------
total    4      0.35       0.38     200          70523       5
```

Şekil 6.7.Nehir üzerinde seçilen node'a en yakın 5 noktanın bulunması

```
17.SELECT /*+ FIRST_ROWS*/ a.objectid FROM river_node rn, agi
a WHERE rn.node_id=173 and a.havzano=21 AND SDO_NN(a.geom,
rn.geometry  ,'SDO_BATCH_SIZE=100')='TRUE'     AND  ROWNUM<=5
ORDER BY a.objectid;

call     count       cpu    elapsed       query    current        rows
------- ------  -------- ----------  ---------- ----------  -------
Parse        1      0.01       0.02        831          0          0
Execute      1      0.00       0.00          0          0          0
Fetch        2      0.29       0.37        173      70523          5
------- ------  -------- ----------  ---------- ----------  -------
total        4      0.31       0.40       1004      70523          5
```

Şekil 6.8.Nehir üzerinde seçilen node'a en yakın 5 noktanın bulunması



Şekil 6.9. En yakın 5 noktanın bulunması sorguları performans sonuçları

Şekil 6.6.'da görünen SQL'in çalışma sistemini incelediğimizde;

Mekânsal indeks en yakın 5 AGİ noktasını döndürecektir ve bu 5 nokta içerisinde havza numarası 21 olan AGİ yoksa diğer en yakın 5 AGİ arasında bu özellikleri sağlayan geometriler bulunacaktır. Bu işlem where cümleciğinde belirtilen şartlar sağlanana kadar iterative bir şekilde devam edecek ve sonuçlar batch'lere gönderilecektir. Batch'lerin boyutları indeks tarafından belirlenmektedir. Eğer en yakın 5 AGİ noktasının ilk 100 kayıt içerisinde olduğunu tahmin edebiliyorsak

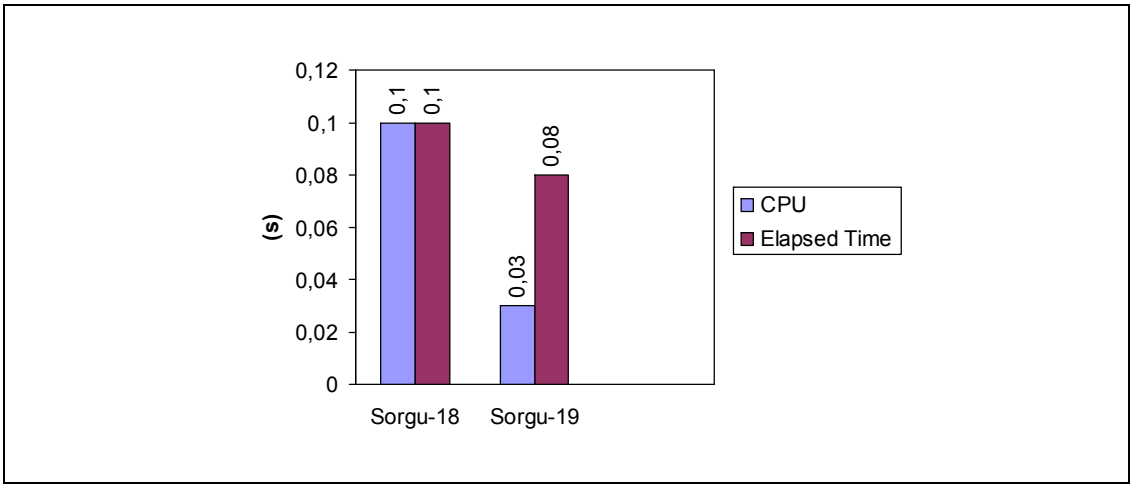
SDO_BATCH_SIZE parametresini 100'e set ederek batch boyutunu belirlenebilmektedir.

Şekil 6.7.'deki SQL'de görüldüğü gibi bu işlem sorgunun çalışma performansını artırmıştır. Şekil 6.8.'deki SQL'de kullanılan FIRST_ROWS hint'i optimizer'ın çalışma moduna müdahale ederek sorgu sonucunda ilk değerlerin getirilmesi istenildiğinde indeks kullanımını tetiklediği için Şekil 6.9.'da görüldüğü gibi sorgu performansını artırmıştır. İndeks kullanımı I/O artışına sebep olsa da performans artışı yaşanmaktadır.

İkinci olarak da SDO_NUM_RES parametresi kullanımı test edilmiştir. Yukarıda test edilen "river_node tablosu üzerinde seçilen bir noktanın 21 no'lu havzada yer alan en yakın 5 AGİ noktasının bulunması" sorgusu inner tablo için bir ölçüt belirtilmeden çalıştırılmak istendiğinde (burada where cümleciğinde havza numarası kullanılmayacaksa) SDO_NUM_RES parametresi ile sorgunun çalıştırılması daha hızlı sonuç elde edilmesini sağlamaktadır. Bu yönde çalıştırılan sorgular Çizelge 6.6.'da görülmektedir. Çalıştırılan sorgular için elde edilen performans değerleri de Şekil 6.10.'da görülmektedir.

Çizelge 6.6. SDO_NN operatörü için "ROWNUM" ve "SDO_NUM_RES" parametreleri kullanımı

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
18.	`SELECT /*+NO_INDEX*/ a.objectid FROM akarsular aks, AGI a WHERE aks.objectid=19541 and a.havzano=8 and SDO_NN(a.geom, aks.geom)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid;`	Seçilen bir Akarsuya en yakın 5 AGİ noktasının tesbiti
19.	`SELECT /*+NO_INDEX*/ a.objectid,a.havzano FROM akarsular aks, AGI a WHERE aks.objectid=19541 and SDO_NN(a.geom, aks.geom,' SDO_NUM_RES=5')='TRUE' ORDER BY a.objectid;`	Seçilen bir Akarsuya en yakın 5 AGİ noktasının tesbiti
20.	`SELECT /*+ ORDERED */ c.objectid,sdo_nn_distance (1) distance_in_miles FROM akarsular aks, agi c WHERE aks.objectid=19495 AND sdo_nn(c.geom, aks.geom, 'sdo_num_res=5 unit=mile', 1) = 'TRUE'ORDER BY c.objectid;`	Seçilen bir Akarsuya en yakın 5 AGİ noktasının tesbiti

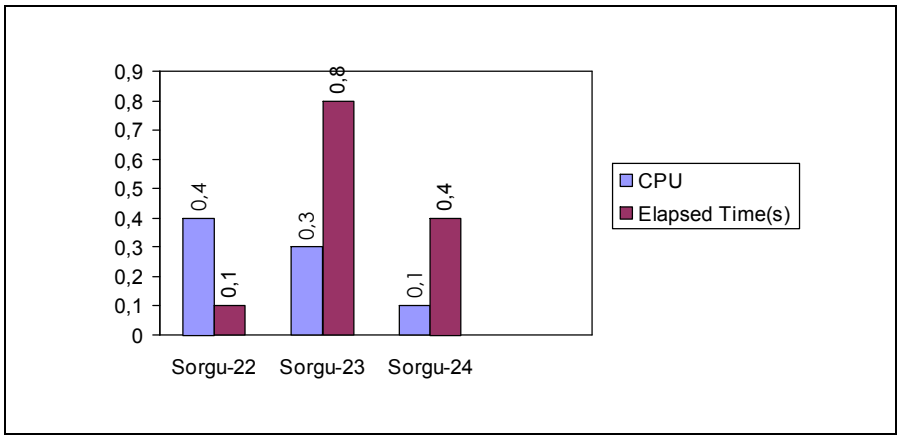


Şekil 6.10. En yakın 5 noktanın bulunması sorguları performans sonuçları

Mekânsal veriler üzerinde indeks yaratılırken Çizelge 6.7.'de 21 numaralı sorgu cümlesinde olduğu gibi geometri kolonuna eklenen verilerin tipini belirtmek mümkündür. İndeks yaratılırken ya da indeks rebuild işlemleri ile "layer_gtype" parametresi tanımlanabilmekte ve çalıştırılan sorgularda çok ciddi performans artışı sağlanmaktadır. Mekânsal indeksi yaratırken "layer_gtype" parametresini set ederek çalıştırdığımız Çizelge 6.7.'de 22, 23 ve 24 sorgu numaralı SQL cümlelerinin çalıştırılması ile elde edilen performans değerleri de Şekil 6.11.'de görülmektedir.

Çizelge 6.7. “LAYER_GTYPE” parametresi ile yaratılan index için test edilen SQL’ler

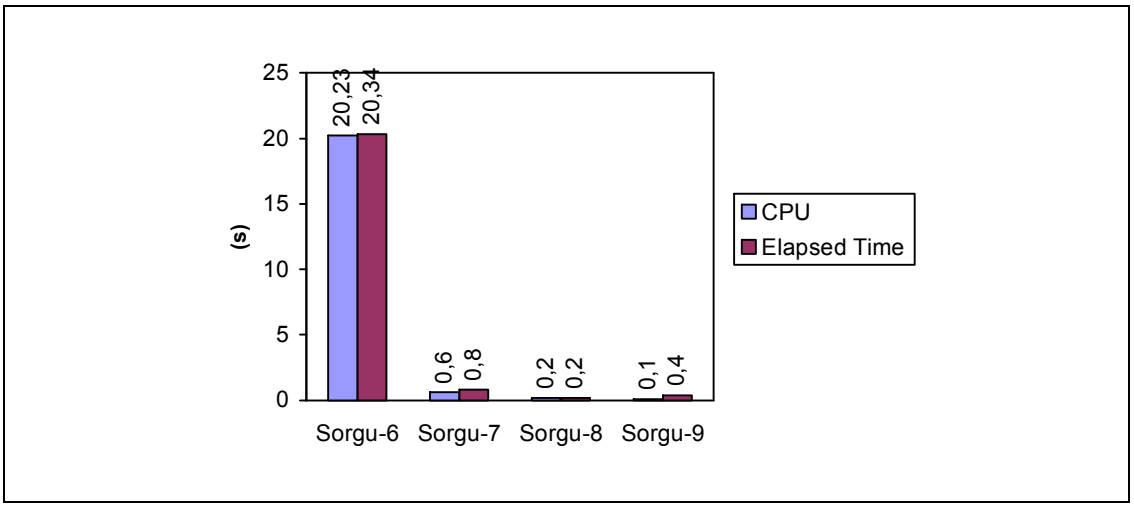
Sorgu No.	Test edilen SQL cümlesi	Açıklama
21.	CREATE INDEX AGI_GLAYER_SPATIAL_IDX ON AGI_GLAYER(GEOM) INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS ('LAYER_GTYPE=POINT');	“LAYER_GTYPE =POINT” parametresi kullanılarak spatial index yaratılması
22.	SELECT a.objectid FROM river_node rn, AGI_GLAYER a WHERE rn.node_id=173 AND SDO_NN(a.geom, rn.geometry)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid ;	Nehir üzerinde seçilen node’a en yakın 5 noktanın bulunması
23.	SELECT a.objectid FROM river_node rn, AGI_GLAYER a WHERE rn.node_id=173 and a.havzano=21 AND SDO_NN(a.geom,rn.geometry ,'SDO_BATCH_SIZE=100')='TRUE' AND ROWNUM<=5 ORDER BY a.objectid;	Nehir üzerinde seçilen node’a en yakın 5 noktanın bulunması
24.	SELECT /*+ FIRST_ROWS */ a.objectid FROM river_node rn, AGI_GLAYER a WHERE rn.node_id=173 and a.havzano=21 AND SDO_NN(a.geom, rn.geometry ,'SDO_BATCH_SIZE=100')='TRUE' AND ROWNUM<=5;	Nehir üzerinde seçilen node’a en yakın 5 noktanın bulunması



Şekil 6.11. En yakın 5 noktanın bulunması sorgularının “LAYER_GTYPE=POINT” parametresi kullanılarak oluşturulan indeks ile çalıştırılması sonucu elde edilen performans sonuçları

Şekil 6.11.'de de görüldüğü gibi Agi_glayer tablosunda tanımlı nokta (point) tipi geometri kolonu üzerinde tanımlı indeks layer_gtype parametresi ile geometri kolonunda sadece point türü veriler tutulması belirtilmiş ve sorguların çalışmasında ciddi performans artışı sağlanmıştır.

Oracle, yaratılan SDO Geometri kolonuna farklı tipte geometrilerin eklenebilmesine imkân sağlamaktadır. Ancak geometri kolonu üzerinde tanımlanan indeks ile bu esneklik kısıtlanabilmektedir. AGI tablosu üzerinde oluşturulan mekansal indeks "LAYER_GTYPE=POINT" şeklinde parametre kullanımı ile tekrar oluşturulmuş, bir önceki bölümde Çizelge 6.3.'de SDO_JOIN operatörü için test edilen SQL cümleleri(6, 7, 8 ve 9 sorgu numaralı SQL cümleleri) tekrar test edilmiştir. Elde edilen sonuçlara ait grafik Şekil 6.12.'de gösterilmiştir. Grafikte de görüldüğü üzere SDO_GEOMETRY kolonu üzerinde bu şekilde oluşturulan indeks ile çalıştırılan SQL performansında ciddi artış olmaktadır.



Şekil 6.12. SQL'lerin "LAYER_GTYPE=POINT" parametresi kullanılarak elde edilen performans sonuçları

Büyük tablolar için partitioned spatial indeks kullanımı bu operatörlerin çalışmasını hızlandırsa da bu tez kapsamında kullanılan tabloların çok büyük tablolar olmaması sebebiyle bu özellik kullanılmamıştır.

6.1.5. Normal İndeks Kullanımında Dikkat Edilmesi Gereken Hususlar

Bir SQL cümlesi içerisinde normal indeks ve mekansal indeks'in bir arada kullanımı yanlış sonuçlar üretilmesine neden olabilmektedir. Bunun önemli sebebi mekansal indekslerin 2 boyutlu olması ve normal (B-tree) indekslerin tek boyutlu olmasıdır. Bu sorunu aşmak için indeks kullanmamak mümkündür. Ancak tablonun farklı analizler için de kullanılabileceğini düşündüğümüzde "NO_INDEX" hint'i ile optmizer'ın "havzano" kolonu üzerinde tanımlanmış olan indeksi kullanmamasını sağlamak daha doğru bir yol olmaktadır.

Bu durumu test etmek amacıyla Çizelge 6.8.'de belirtilen SQL'ler ile "Fırat nehri üzerinde seçilen bir noktaya en yakın Akım Gözlem İstasyonlarının bulunması" sorgusu "havzano" kolonunda normal indeks kullanılıp kullanılmamasına göre test edilmiştir. SDO_NN operatörü, sorgu tablosu(query_table)'nda bulunan "havzano" kolonu üzerinde indeks tanımlı olması ve bu kolonunun where cümleciği içinde kullanılması durumunda hatalı sonuç üretmektedir. Çizelge 6.8.'de 25. SQL'de "havzano" kolonu üzerinde indeks olmadığı durumda elde edilen doğru değerler görülmektedir. 26 numaralı SQL'de "havzano" kolonunda indeks'in olması durumu ve bu kolonun da where cümleciğinde kullanılması sonucunda elde edilen yanlış değerler görülmektedir. 27 sorgu numaralı SQL'de ise "havzano" kolonunda indeks olduğu ve NO_INDEX hint'i kullanıldığı durum görülmektedir. NO_INDEX hint'i ile "havzano" kolonu üzerinde bulunan indeksin kullanılmaması sağlanarak doğru sonuçlar elde edilmiştir.

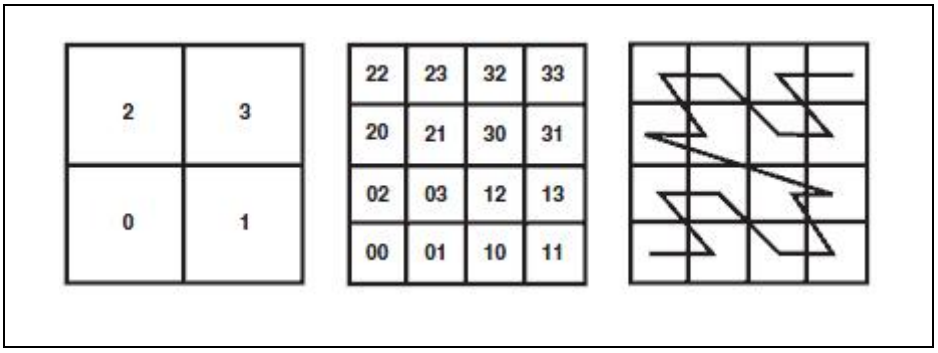
Çizelge 6.8. Spatial Operatörlerin normal index ile kullanımı

Sorgu No.	Test edilen SQL cümlesi	Açıklama
25.	SELECT a.objectid FROM agi a ,river_node rn WHERE rn.node_id=173 and ct.havz ano=21 AND SDO_NN(a.geom, rn.geometry)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid ; OBJECTID ---------- 1683 1958 1965 1971 2067	Seçilen node'a en yakın 5 AGİ'nin bulunması
26.	SELECT a.objectid FROM agi a ,river_node rn WHERE rn.node_id=173 and a.havzano=21 AND SDO_NN(a.geom, rn.geometry)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid ; OBJECTID ---------- 276 277 278 279 280	Seçilen node'a en yakın 5 AGİ'nin bulunması
27.	SELECT /*+NO_INDEX(a AGI_HAVZANO_IDX)*/ a.objectid FROM agi a ,river_node rn WHERE rn.node_id=173 and ct.havzano=21 AND SDO_NN(a.geom, rn.geometry)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid ; OBJECTID ---------- 1683 1958 1965 1971 2067	seçilen node'a en yakın 5 AGİ'nin bulunması

6.2. Q-Tree index Kullanımı ve Sonuçları

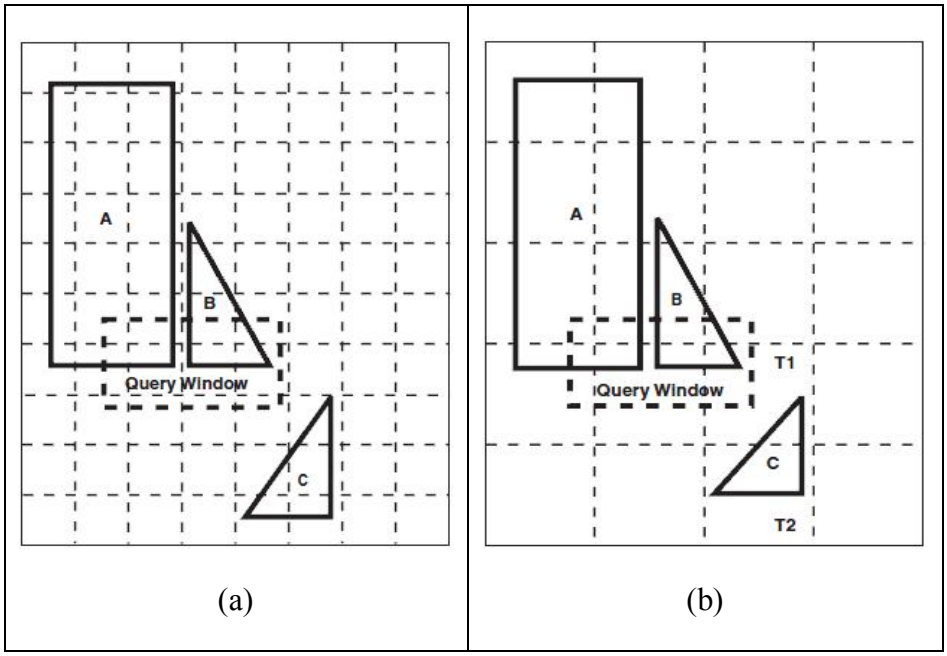
6.2.1. Q-tree index Kavramı

Quadtree indeks ya da kısaca q-tree indeks Şekil 6.13'de de görüldüğü üzere geometri alanı, yani bütün geometrilerin bulunduğu katman tesselation adı verilen bir işleme tabi tutularak her geometri için özel ve ayrıntılı tile'lar elde edilir. Bu tile'lara z-order ya da Morton kodu adı verilen metod ile tek (unique) bir sayısal değer atanır.



Şekil 6.13. Quadtree Ayrıştırma ve Morton Kodlaması

Oracle iki boyutlu mekansal veriler için q-tree indeks kullanımına izin vermektedir. Q-tree indeksler için tile boyutu seçimi dikkatli yapılması gereken ve sorgu performansı için de önemli bir işlemdir. Şekil 6.14.(a)'da tile'ların daha küçük seçilmesi sebebi ile "C" geometrisi sorguya katılmazken Şekil 6.14. (b)'de "C" geometrisi sorguya dâhil edilerek analizler yapılmaktadır. Tile'ların küçük seçilmesi performansı düşüren bir faktör olmasına rağmen, kesin sonuçların elde edilmesi için önemlidir. Tile'lar her bir geometri için "interior" ya da "boundary" şeklinde etiketlenmektedir. Bir geometrinin sınırının herhangi bir bölümü tile'a dokunursa tile durumu "boundary", eğer tile geometrinin tamamiyle içinde ise tile durumu "interior" olarak etiketlenmektedir. Bu tür etiketleme diskte çok küçük bir alan kaplamakla birlikte, sorgu performansını etkilememektedir[20].



Şekil 6.14. Küçük ve Büyük boyutlu Tile ile Geometri Gösterimi [45]

Oracle coğafi koordinat sisteminde tanımlı veri üzerinde üzerinde q-tree indeks yaratılmasına imkân vermemekle birlikte, aşağıda görüldüğü gibi coğrafi veri üzerinde de q-tree indeks oluşturulabilmektedir. Ancak coğrafi koordinat sisteminde tanımlı veriler üzerinde indeks oluşturulması SDO_WITHIN_DISTANCE operatörünün kullanımını engellemektedir.

```
CREATE INDEX HAVZALAR_Q_SIDX ON HAVZALAR_Q(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('geodetic=FALSE SDO_LEVEL=8');
```

Daha önceki bölümlerde anlatılan SDO_NN operatörü için SDO_BATCH_SIZE parametresi de q-tree indeks için kullanılamamaktadır.

Coğrafi koordinat bilgisine sahip makansal verinin mesafe, bölge ve açısal bilgiye sahip olması, üzerinde yapılan analizlerin kesin ve tüm yerkürenin modellenmesini sağlamaktadır. Bu bilgilerin q-tree indekste olmaması tüm yerkürenin modellenmesini engellemektedir.

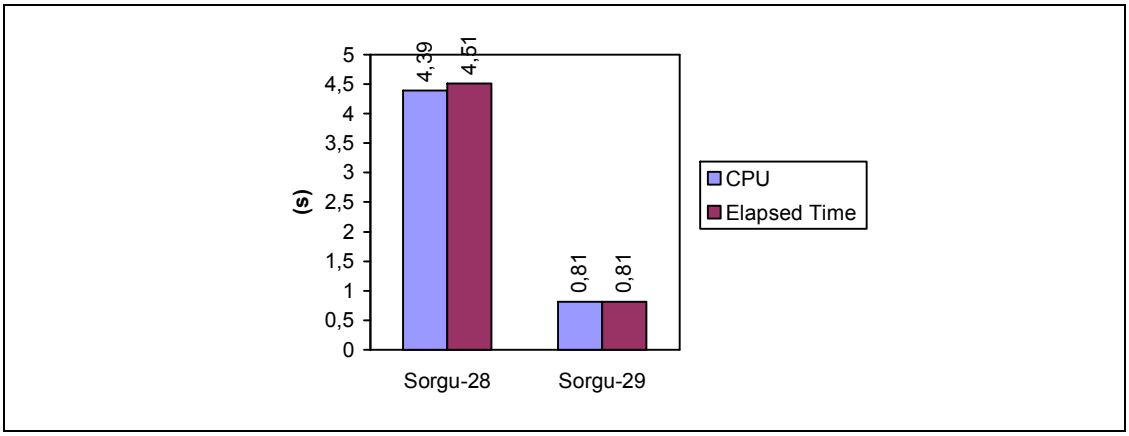
Yukarıda bahsedilen özellikler de göz önüne alınarak, bu çalışmada kullanılan veriler üzerinde q-tree indeks yerine r-tree indeks kullanımı tercih edilmiştir. Ancak kullanımı bazı durumlar için önerilen q-tree indeksin bu çalışmada kullanılan veriler üzerinde tanımlanarak, yukarıda performans testleri yapılan bazı SQL cümleleri ile testi ve sonuçları aşağıda açıklanmıştır.

6.2.2. Belirli Bir mesafe içinde kalan geometrilerin tesbiti

Yukarıdaki bölümlerde bir geometriye belirli bir mesafe içerisinde kalan geometrilerin bulunması amacıyla mekansal veriler üzerinde r-tree indeks tanımlanarak SDO_WITHIN_DISTANCE operatörü test edildi. Bu bölümde q-tree indeks kullanılarak aynı SQL cümleleri test edilmiştir. Çalıştırılan sorgular ve test sonuçları aşağıda gösterilmiştir. Çalıştırılan SQL cümleleri sonucu Şekil 6.14'de görülmektedir.

Çizelge 6.9. Q-tree indeksin test edilmesi amacıyla SDO_WITHIN_DISTANCE operatörü için denenen SQL cümleleri

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
28.	`SELECT distinct C.objectid FROM akarsu_q R, agi_q C WHERE R.Akarsu_ad='Firat N.' AND SDO_WITHIN_DISTANCE(C.geom,R.geom, 'DISTANCE=0.05')='TRUE' order by C.objectid ;`	Q-tree indeks kullanılarak Fırat Nehrine seçilen node'a 0.05 birim mesafede bulunan AGİ'lerin bulunması
29.	`SELECT distinct C.objectid FROM akarsu R,agi C WHERE R.Akarsu_ad='Firat N.' AND SDO_WITHIN_DISTANCE(C.geom, R.geom, ' DISTANCE=5 UNIT=KM')='TRUE' order by C.objectid ;`	R-tree indeks kullanılarak Fırat Nehrine 5 km. mesafede bulunan AGİ'lerin bulunması



Şekil 6.15. Q-tree ve R-tree indeks ile çalıştırılan Çizelge 6.9. SQL'lerinin performans değerleri

Yukarıdaki grafikte de görüldüğü üzere SDO_WITHIN_DISTANCE operatörünün q-tree indeks ile kullanımı performans açısından uygun olmamaktadır.

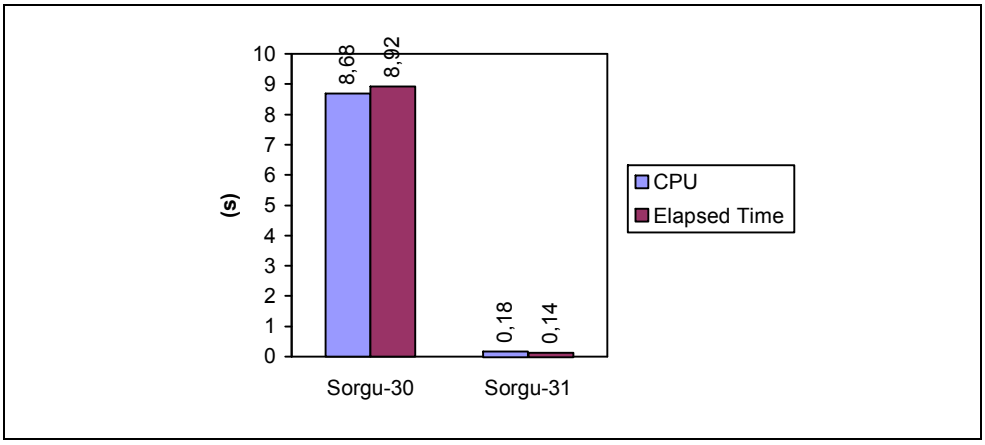
6.2.3. Bir geometriye en yakın geometrilerin tesbiti

Bu sorgu için SDO_NN operatörü kullanılmış olup, 8. havzaya ait nehirlere en yakın 5 AGİ noktasının tesbiti için kullanılan sorgular Çizelge 6.10.'da görülmektedir.

Çizelge 6.10. Q-tree ve R-tree indeks kullanılarak seçilen havzaya ait nehirlere en yakın 5 AGİ noktasının tesbiti

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
30.	`SELECT /*+NO_INDEX*/ a.objectid FROM akarsu_q aks, AGI_q a WHERE a.havza_no='08' and SDO_NN(a.geom,aks.geom, 'SDO_LEVEL=9')='TRUE' AND ROWNUM<=5 ;`	Q-tree index kullanılarak 8. havzaya ait nehirlere en yakın 5 AGİ noktasının tesbiti
31.	`SELECT /*+NO_INDEX*/ a.objectid FROM akarsular aks, AGI a WHERE a.havza_no='08' and SDO_NN(a.geom, aks.geom)='TRUE' AND ROWNUM<=5 ORDER BY a.objectid;`	R-tree index kullanılarak 8. havzaya ait nehirlere en yakın 5 AGİ noktasının tesbiti

Çizelge 6.10.'de görünen sorgular çalıştırılarak elde edilen sonuçlar Şekil 6.15'de gösterilmiştir. Sonuç olarak, R-tree indeks kullanılarak tasarlanan sorguların çalıştırılması ile elde edilen performans sonuçlarının daha iyi olduğu görülmüştür.



Şekil 6.16. Q-tree ve R-tree indeks ile çalıştırılan Çizelge 6.10. SQL'lerinin performans değerleri

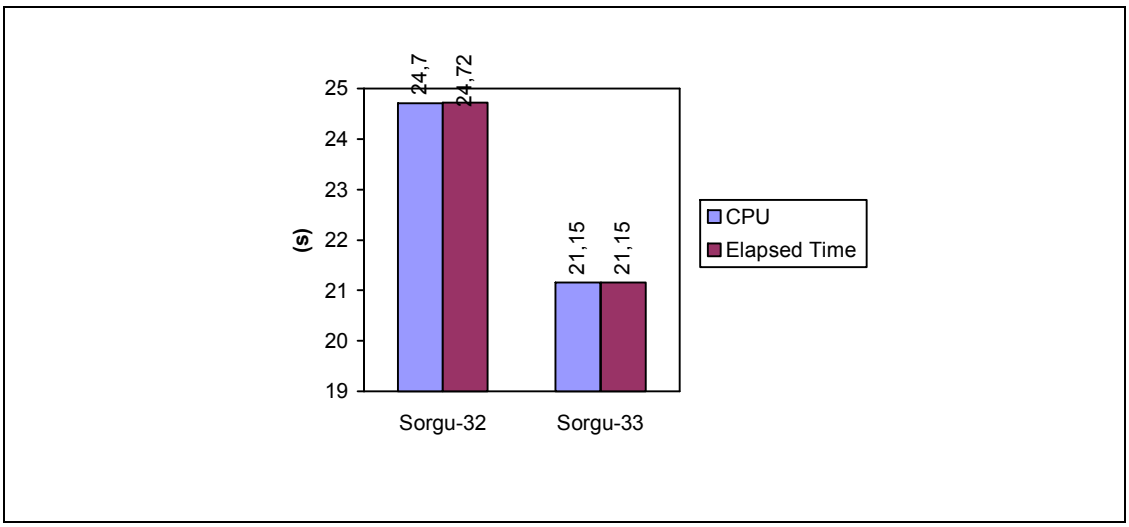
Q-tree indeks tanımlanmış veri üzerinde kullanılan SDO_NN operatörü için R-tree indeks ile kullanabildiğimiz sdo_batch_size parametresi kullanılamamaktadır.

6.2.4. Seçilen geometri içinde kalan geometrilerin tesbiti

Yukarıdaki bölümlerde R-tree indeks ile denenmiş olan "Havzalar içinde kalan akarsular sorgusu" gibi burada da q-tree indeks için "Havzalar içinde kalan Agi'lerin tesbiti" sorgusu denenmiştir. Test edilen SQL cümleleri Çizelge 6.11.'de görülmektedir. Çizelge 6.11.'de çalıştırılan SQL cümlelerinin sonuçları Şekil 6.17'de gösterilmektedir.

Çizelge 6.11. Q-tree ve R-tree indeks kullanılarak seçilen Havza içinde kalan Akarsuların tesbiti

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
32.	`SELECT /*+ ORDERED */ distinct a.objectid FROM havzalar_q b, akarsu_q a WHERE b.havza_no='21' and SDO_RELATE(a.geom, b.geom,'mask=inside') = 'TRUE';`	Q-tree index kullanılarak Havzalar içinde kalan Akarsuların tesbiti
33.	`SELECT /*+ ORDERED */ distinct a.objectid FROM havzalar b, akarsular a WHERE b.havza_no='21' and SDO_RELATE(a.geom, b.geom,'mask=inside') = 'TRUE';`	R-tree index kullanılarak Havzalar içinde kalan Akarsuların tesbiti



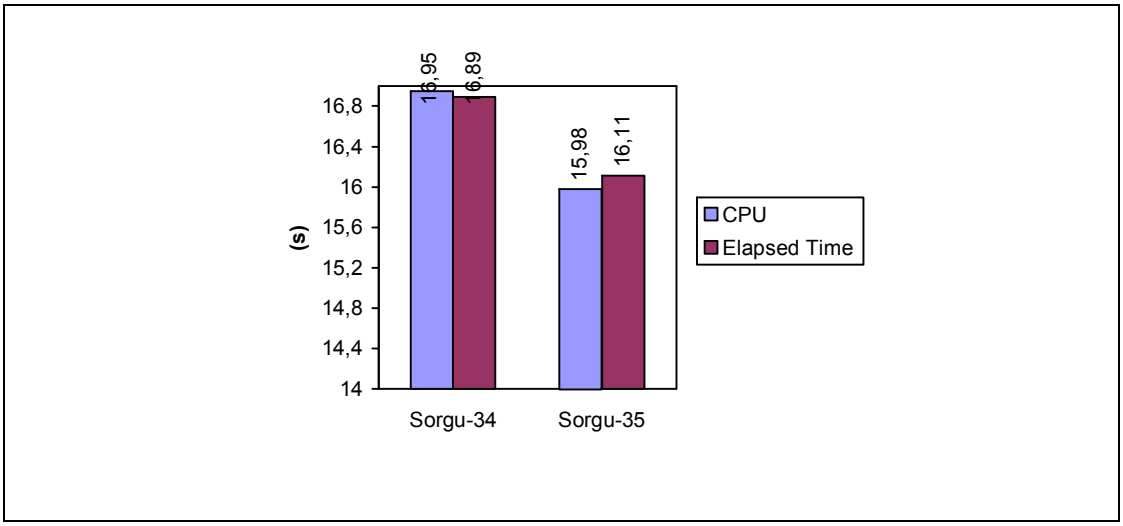
Şekil 6.17. Q-tree ve R-tree indeks ile çalıştırılan Çizelge 6.11. SQL'lerinin performans değerleri

Buraya kadar yapılan çalışmalar sonucunda Q-tree indekslerin kullanıldığı veriler ile test edilen SQL cümlelerinin performanslarının R-tree indeks kullanılan sorgulara göre daha düşük olduğu görülmüştür. Ancak q-tree indekslerin avantaj sağladığı durumlar da bulunmaktadır. Yukarıda da açıklandığı gibi q-tree indekslerin durum bilgileri tile'ın içinde kalma ya da geometri sınırlarının tile'a dokunmasına göre belirlenmekte. R-tree indekslerde ise geometri MBR'ler şeklinde ifade edilerek indeks yaratılmaktadır. Bu durumun testi amacıyla iki geometrinin birbirine dokunmasını ifade eden "touch" parametresi ile tasarlanan sorgular Çizelge 6.12.'de görülmektedir.

Çizelge 6.12. Q-tree ve R-tree indeks kullanılarak birbirine dokunan Havzaların tesbiti

Sorgu No.	**Test edilen SQL cümlesi**	**Açıklama**
34.	`SELECT a.objectid, b.OBJECTID FROM havzalar_q b, havzalar_q a WHERE SDO_RELATE(a.geom, b.geom, 'MASK=TOUCH' )='TRUE' ;`	Q-tree index kullanılarak birbirine dokunan Havzaların tesbiti
35.	`SELECT a.objectid, b.OBJECTID FROM havzalar b, havzalar a WHERE SDO_RELATE(a.geom, b.geom, 'MASK=TOUCH' )='TRUE' ;`	R-tree index kullanılarak birbirine dokunan Havzaların tesbiti

Çizelge 6.14.'de görünen SQL'lerin çalıştırılması sonucu Şekil 6.18.'de görünen grafik elde edilmiştir. SDO_WITHIN_DISTANCE ve SDO_NN düşünüldüğünde touchparametresi ile tasarlanan sorguların daha iyi performans gösterdiği görülmektedir.



Şekil 6.18. Q-tree ve R-tree indeks ile çalıştırılan Çizelge 6.12. SQL'lerinin performans değerleri

6.3. Sonuçlar

Bu bölümde, çalışma kapsamında R-tree indeks seçiminde rol oynayan nedenler yapılan performans testleri ile açıklanmıştır. Çalışmada kullanılan verilerin coğrafi koordinat sisteminde tanımlı olması, ihtiyaç duyulan sorguların gerektirdiği operatörlerin R-tree indeks ile kullanılabilir olması, q-tree indekslerin tuning işleminin pahalı işlemler olması ve q-tree indekslerle tüm yerkürenin modellenememesi gibi sebeplerden dolayı R-tree indeks tercih edilmiştir. Ayrıca Oracle da performans açısından R-tree indeks kullanımını salık vermektedir.

BÖLÜM 7

SONUÇLAR VE DEĞERLENDİRME

Bu çalışmada coğrafi verilerin sorgulanması amacıyla Open Geospatial Consortium-OGC ve World Wide Web Consortium-W3C tarafından oluşturulmuş açık standartlar kullanılarak web uygulaması geliştirilmiştir. Çalışmada Coğrafi veriler, Oracle Spatial veritabanında depolanarak oracle'ın getirdiği operatörlerin kullanılması ile çok karmaşık olabilecek sorgular SQL cümleleri ile rahatlıkla tasarlanmıştır. Bu çalışmada mekansal verilerin sorgulanmasında kullanılan SQL cümleleri tasarlanırken dikkat edilecek hususlara değinilmiş, yardımcı araçların da kullanılmasıyla iyileştirme çalışmaları yapılmıştır.

Veritabanında geliştirilen Stored Prosedürler(Saklı Yordam) ile uygulama üzerinden alınan parametreler(Havza numarası, Baraj Id v.b.) yardımı ile görüntülenmek istenen mekansal veriye ilişkin XML ve KML dosyaları üretilmiştir. Açık kaynak kodlu ve internet üzerinden de rahatlıkla elde edilebilecek Openlayers Api'si ile yine internetten rahatlıkla ulaşılabilen Google Maps Api'si kullanılarak uygulama üzerinden bu XML ve KML dosyaları görüntülenmiştir. Kullanılan servisler yardımı ile tasarlanan sistem farklı sistemlerle rahatlıkla entegre edilebilmektedir.

Mekânsal verilerin görüntülenmesini ve editlenmesini sağlayan Java tabanlı bir sunucu yazılımı olan GeoServer kullanılmış, GeoServer üzerinden bir OGC standartı olan WMS servisi kullanılarak Nehir, Akarsu ve Havza gibi coğrafi veriler uygulamaya katman olarak eklenmiştir.

Bu çalışma ile mekansal verilerin internet tarayıcısı üzerinden yayımlanması amacıyla yaygın olarak kullanılan mimariye örnek teşkil edecek bir yapı detaylı olarak anlatılmış, geliştirilen SQL cümleleri performans iyileştirme araçları da kullanılarak sisteme entegre edilmiştir.

Çalışmada kullanılan coğrafi veriler üzerinde çeşitli sorgular denenerek veritabanı performans değerleri incelenmiştir. Veritabanı parametrelerinde de değişiklikler

yapılmakla birlikte bu çalışma kapsamında coğrafi sorguların tasarlanması öncelikli olarak incelenmiştir. Oracle spatial teknolojisi ile çok karmaşık sorgular SQL cümleleri şeklinde kolaylıkla yazılabilmiş, kullanılan verilerin özelliklerine göre operatörler seçilmiş, birincil filtre kullanımının tercih edilmesinin avantaj ve dezavantajları, spatial operatörlerin spatial fonksiyonlara tercih edilme sebepleri gibi konulara cevap verilmiştir.

Çalışma kapsamında oluşturulan Network Data Model yapısı geometrik verilerin yönlü çizge şeklinde ifade edilmesi sağlanmış, nehrin aşağı ve yukarısı şeklinde tanımlanabilen sorguların yazılması bu yolla sağlanmıştır. Bu çalışma ileride yapılacak projeler için başlangıç oluşturması ve Oracle Spatial Topoloji Data Model kullanılarak yeni çalışmaların yapılmasına altyapı oluşturması sebebiyle önem arz etmektedir.

Bundan sonraki çalışmalarda, mekansal verilerin mobil uygulamalar ile kullanılması ve doğal felaket durumlarında risk yönetimine yardımcı projelerin üretilmesi planlanmaktadır. Yapılacak çalışmalarda uzaktan algılama teknolojileri de kullanılarak vektör veriler ile uydu verilerinin bir arada kullanıldığı çalışmalar planlanmaktadır.

KAYNAKLAR

[1] Ravi Kothuri, Albert Godfrind, Euro Beinat, Pro Oracle Spatial , Apress 2004

[2] Oracle® Spatial User's Guide and Reference 10g Release 2 (10.2), 2005

[3] Oracle® DatabaseAdministrator's Guide 10g Release 2 (10.2), 2006

[4] Oracle® Spatial Topology and Network Data Models 10g Release 2 (10.2), 2005

[5] Gavin Powell, Oracle® Performance Tuning for 10gR2 Second Edition, Elsevier Digital Press, 2007

[6] Sam R. Alapati, Expert Oracle Database 10g Administration, Apress, 2005

[7] Improving Performance using Query Rewrite in Oracle Database 10g An Oracle White Paper December 2003

[8] Linda McKinnon Al McKinnon, XML in 60 Minutes a Day , Wiley Publishing, 2003,

[9] Thiru Thangarathinam, Professional ASP.NET 2.0 XML, Wiley Publishing, 2006

[10] Brian Benz with John R. Durant, XML Programming Bible, Wiley Publishing, 2003

[11] Michael Purvis Jeffrey Sambells Cameron Turner, Beginning Google Maps Applications with PHP and Ajax, Apress, 2006

[12] Scott Davis , GIS for Web Developers, The Pragmatic Bookshelf Raleigh, North Carolina Dallas, Texas, 2007

[13] Ron Lake David Burggraf Laurie , Geography Mark-Up Language : Foundation for the Geo-Web , John Wiley & Sons, 2004

[14] Josie Wernecke, The KML Handbook: Geographic Visualization for the Web, Addison-Wesley, 2008

[15] Oracle® Spatial Quadtree Indexing 10g Release 1 (10.1), December 2003

[17] Advanced Spatial Data Management for Enterprise Applications, An Oracle White Paper January 2009

[18] Empowering Applications with Spatial Analysis and Mining, An Oracle White Paper January 2009

[19] Thomas Kyte, Expert Oracle Database Architecture 9i and 10g Programming Techniques and Solutions, Apress, 2005

[20] Ravi Kant V Kothuri, Siva Ravada, Daniel Abugov Quadtree and R-tree Indexes in Oracle Spatial : A Comparison using GIS Data , SIGMOD '2002 June 4-6, Madison, Wisconsin, USA

[21] HANAN SAMET , A Sorting Approach to Indexing Spatial Data, Center for Automation Research Institute for Advanced Computer Studies Computer Science DepartmentUniversity of Maryland College Park, Maryland 20742, USA, May 18, 2008

[22] Kian-Lee Tan, Member, Beng Chin Ooi and David J. Abel, Exploiting Spatial Indexes for Semijoin-Based Join Processing in Distributed Spatial Databases, IEEE Transactions on Knowledge and Data Engineering, Vol. 12, NO. 6, DECEMBER 2000

[23] Thoinus Brinkhoff Hans-Peter Kriegel Bernhard Seeger , Efficient Processing of Spatial Joins Using R-trees Institute of Computer Science, University of Munich Leopoldstr. 11 B, W-8000 Munchen 40, Germany
[24] Guttman A.: 'R-trees: A Dynamic Index Structure for S atial Searching', Proc. ACM SIG MOD Int. Coaf. on Management o! Data, Boston,MA., 1984,
[25] Nikos Mamoulis, Dimitris Papadias, Integration of Spatial Join Algorithms for Processing Multiple Inputs SIGMOD '99 Philadelphia PA Copyright ACM 1999
[26] K. V. Ravi Kanth, Siva Ravada, Jayant Sharma and Jay Banerjee, Indexing Medium-dimensionality Data in Oracle, SIGMOD '99 Philadelphia PA Copyright ACM 1999
[27] Ning An, Ravi Kanth V Kothuri, Siva Ravada, Improving Performance with Bulk-Inserts in Oracle R-Trees Proceedings of the 29th VLDB Conference,Berlin, Germany, 2003
[28] Diane Greene ,An Implementation and Performance Analysis of Spatial Data Access Methods Computer Sciences Department University of Califomia, Berkeley Berkelev. Ca., IEEE, 1989
[29] Maruto Masserie Sardadi, Mohd Shafry bin Mohd Rahim, Zahabidin Jupri, and Daut bin Daman, Choosing R-tree or Quadtree Spatial Data Indexing in One Oracle Spatial Database System to Make Faster Showing Geographical Map in Mobile Geographical Information System Technology, World Academy of Science, Engineering and Technology 46 2008
[30] Tapu ve Kadastro Bilgi Sistemi Projesinde Kadastral Verilerin Yönetimi O.Mataracı
[31] TMMOB Harita ve Kadastro Mühendisleri Odası Ulusal Cografi Bilgi Sistemleri Kongresi 30 Ekim –02 Kasım 2007, KTÜ, Trabzon Coğrafi Bilgi Sistemleri Birlikte Çalışabilirlik Esasları E.Boza
[32] Birol ALAS, Doğan UÇAR Coğrafi işaretleme dilinin tapu ve kadastro verileri için sanal doku ortamında kullanılması itüdergisi/d mühendislik Cilt:7, Sayı:6, 36-46 Aralık 2008
[33] http://mygeoraptor.googlepages.com/home (Son Erişim Tar:26.11.2009)
[34] http://www.oracle.com/technology/products/database/sql_developer/ (Son Erişim Tar:26.11.2009)
[35] http://www.postgresql.org/docs/faqs.FAQ_turkish.html#1.1 (Son Erişim Tar:26.11.2009)
[36] http://www.postgis.org/ (Son Erişim Tar:26.11.2009)
[37] http://www.esriturkiye.com.tr/Turkish/Sayfa.asp?SayfaID=87 (Son Erişim Tar:26.11.2009)
[38] http://mapserver.org/ (Son Erişim Tar:26.11.2009)
[39] http://java.com/tr/download/faq/java_javascript.xml (Son Erişim Tar:26.11.2009)
[40] http://www.w3schools.com/js/default.asp (Son Erişim Tar:26.11.2009)
[41] http://www.esri.com/ (Son Erişim Tar:26.11.2009)
[42] http://download.oracle.com/docs/cd/B10501_01/server.920/a96533/ optimops.htm (Son Erişim Tar:26.11.2009)
[43] OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1

[44] http://narencoolgeek.blogspot.com/2008/01/difference-between-cpu-time-and-elapsed.html
[45] Oracle® Spatial Quadtree Indexing 10g Release 1 (10.1)
[46] http://openlayers.org/
[47] http://www.w3c.org/

EK A

```
cmd>shp2sdo.exe agi agi -g geom -d -x (-180,180) -y (-90,90) -s 8307
-t 0.5 -vspatial

sql>sqlplus   spatialndx/123@tumay   @agi.sql(agi.sql'i   elle   de
çalistirabilirsin)

cmd>sqlldr   spatialndx/123@tumay   control=agi.ctl(üretilen   agi.ctl
dosyasi veritabaninda AGI

sql>CREATE INDEX agi_spatial_idx ON AGI(GEOM)

INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- agi.sql
--
--   This script creates the spatial table.
--
--   Execute this script before attempting to use SQL*Loader
--   to load the agi.ctl file.
--   This script will also populate the USER_SDO_GEOM_METADATA
table.
--   Loading the .ctl file will populate the AGI table.
--
--   To load the .ctl file, run SQL*Loader as follows
--   substituting appropriate values:
--       sqlldr username/password agi.ctl
--
--   After the data is loaded in the AGI table, you should
--   migrate polygon data and create the spatial index
--
-- Creation Date : Thu Jul 02 09:59:46 2009
-- Copyright 1999, 2004 Oracle Corporation
-- All rights reserved
--
DROP TABLE AGI;

CREATE TABLE AGI (  OBJECTID  NUMBER,
  IST_TIP   VARCHAR2(4),
  BOLGE_NO  VARCHAR2(2),
  HAVZA_NO  VARCHAR2(2),
  IST_NO    VARCHAR2(10),
  NEHIR_AD  VARCHAR2(75),
  IST_AD    VARCHAR2(75),
  DURUMU    VARCHAR2(2),
  IST_AC    NUMBER,
  IST_KAP   NUMBER,
  Y_IST_AC  NUMBER,
  ALAN      NUMBER,
  IST_KOTU  NUMBER,
  KALITE    VARCHAR2(1),
  SEDIMENT  VARCHAR2(1),
  CKS_BOYLAM      VARCHAR2(15),
  CKS_ENLEM       VARCHAR2(15),
```

```
  UTMX      NUMBER,
  UTMY      NUMBER,
  UTM_ZONE  NUMBER,
  Z25_PAFNO       VARCHAR2(10),
  Z100_PAFNO      VARCHAR2(10),
  NOTLAR    VARCHAR2(254),
  GEOM      MDSYS.SDO_GEOMETRY);
DELETE FROM USER_SDO_GEOM_METADATA
  WHERE TABLE_NAME = 'AGI' AND COLUMN_NAME = 'GEOM' ;
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
VALUES ('AGI', 'GEOM',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', -180.000000000, 180.000000000,
0.500000000),
MDSYS.SDO_DIM_ELEMENT('Y', -90.000000000, 90.000000000, 0.500000000)
),
8307);
COMMIT;
```

Ek B

```
Execution Plan
----------------------------------------------------------
Plan hash value: 2057530097

-------------------------------------------------------------------------------

| Id  | Operation                     | Name               | Rows  | Bytes |Temp

Spc| Cost (%CPU)| Time     |

-------------------------------------------------------------------------------

-----------------------------

|   0 | SELECT STATEMENT              |                    | 58429 |  9015K|   |  2049
(1)| 00:00:25 |

|   1 |  SORT ORDER BY                |                    | 58429 |  9015K|20M|  2049
(1)| 00:00:25 |

|   2 |   NESTED LOOPS                |                    | 58429 |  9015K|   |     4
(0)| 00:00:01 |

|   3 |    TABLE ACCESS BY INDEX ROWID| RIVER_NODE         |     1 |    34 |   |     2
(0)| 00:00:01 |

|*  4 |     INDEX RANGE SCAN          | RNODE_NODEID_IDX   |     1 |       |   |     1
(0)| 00:00:01 |

|   5 |    TABLE ACCESS BY INDEX ROWID| AGI                | 58429 |  7075K|   |     4
(0)| 00:00:01 |

|*  6 |     DOMAIN INDEX              | AGI_SPATIAL_IDX    |       |       |   |
|           |

-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("RN"."NODE_ID"=173)
   6 - access("MDSYS"."SDO_WITHIN_DISTANCE"("A"."GEOM","RN"."GEOMETRY",'querytyp

e=FILTER

              DISTANCE=30 UNIT=KM')='TRUE')

Statistics
----------------------------------------------------------
        797  recursive calls
          4  db block gets
      13362  consistent gets
          0  physical reads
          0  redo size
     183117  bytes sent via SQL*Net to client
       9031  bytes received via SQL*Net from client
        788  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      11797  rows processed


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.03          0        786          2           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      788      0.09       0.10          0      12159          2       11797
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      790      0.09       0.13          0      12945          4       11797
```

Ek C

```
Execution Plan
----------------------------------------------------------
Plan hash value: 2057530097

-------------------------------------------------------------------------------

| Id  | Operation                     | Name               | Rows  | Bytes |Temp

Spc| Cost (%CPU)| Time     |

-------------------------------------------------------------------------------

|   0 | SELECT STATEMENT              |                    | 58429 |  9015K|   |
2049   (1)| 00:00:25 |

|   1 |  SORT ORDER BY                |                    | 58429 |  9015K|20M|
2049   (1)| 00:00:25 |

|   2 |   NESTED LOOPS                |                    | 58429 |  9015K|   |
4   (0)| 00:00:01 |

|   3 |    TABLE ACCESS BY INDEX ROWID| RIVER_NODE         |     1 |    34 |   |
2   (0)| 00:00:01 |

|*  4 |     INDEX RANGE SCAN          | RNODE_NODEID_IDX   |     1 |       |   |
1   (0)| 00:00:01 |

|   5 |    TABLE ACCESS BY INDEX ROWID| AGI                | 58429 |  7075K|   |
4   (0)| 00:00:01 |

|*  6 |     DOMAIN INDEX              | AGI_SPATIAL_IDX    |       |       |   |
|          |

-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("RN"."NODE_ID"=173)
   6 - access("MDSYS"."SDO_WITHIN_DISTANCE"("A"."GEOM","RN"."GEOMETRY",'DISTANCE

=30

              UNIT=KM')='TRUE')

Statistics
----------------------------------------------------------
        839  recursive calls
          4  db block gets
      13442  consistent gets
          0  physical reads
          0  redo size
     183117  bytes sent via SQL*Net to client
       9031  bytes received via SQL*Net from client
        788  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
      11797  rows processed
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.04       0.05          0        771          2           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      788      0.14       0.13          0      12159          2       11797
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      790      0.18       0.18          0      12930          4       11797
```

Ek D

```
CREATE TABLE RIVER_NODE
(
  NODE_ID       NUMBER,
  NODE_NAME     VARCHAR2(200 BYTE),
  NODE_TYPE     VARCHAR2(200 BYTE),
  ACTIVE        VARCHAR2(1 BYTE),
  PARTITION_ID  NUMBER,
  GEOMETRY      MDSYS.SDO_GEOMETRY,
  COST          NUMBER
)



CREATE TABLE RIVER_LINK
(
  LINK_ID        NUMBER,
  LINK_NAME      VARCHAR2(200 BYTE),
  START_NODE_ID  NUMBER                          NOT NULL,
  END_NODE_ID    NUMBER                          NOT NULL,
  LINK_TYPE      VARCHAR2(200 BYTE),
  ACTIVE         VARCHAR2(1 BYTE),
  LINK_LEVEL     NUMBER,
  GEOMETRY       MDSYS.SDO_GEOMETRY,
  COST           NUMBER,
  BIDIRECTED     VARCHAR2(1 BYTE)

      )



CREATE TABLE RIVER_PATH
(
  PATH_ID        NUMBER,
  PATH_NAME      VARCHAR2(200 BYTE),
  PATH_TYPE      VARCHAR2(200 BYTE),
  START_NODE_ID  NUMBER                          NOT NULL,
  END_NODE_ID    NUMBER                          NOT NULL,
  COST           NUMBER,
  SIMPLE         VARCHAR2(1 BYTE),
  GEOMETRY       MDSYS.SDO_GEOMETRY
)
CREATE TABLE RIVER_PATH_LINK
(
  PATH_ID  NUMBER                                NOT NULL,
  LINK_ID  NUMBER                                NOT NULL,
  SEQ_NO   NUMBER
)
```

Ek E

```
SQL> ALTER SYSTEM SET TIMED_STATISTICS = TRUE;-- Add timed
statistics to trace file

SQL> C:\orcl\product\10.2.0\db_1\RDBMS\ADMIN\utlxplan.sql

SQL> ALTER SESSION SET SQL_TRACE = TRUE;-- Start Session Tracing

SQL> SELECT hvz.OBJECTID, hvz.HAVZANO

    FROM havzalar hvz, agi a

    WHERE hvz.OBJECTID=1

    AND SDO_WITHIN_DISTANCE

    (a.geom, hvz.geom, 'DISTANCE=0.25 UNIT=MILE ' )='TRUE'

    ORDER BY hvz.OBJECTID;

SQL>

SQL> ALTER SESSION SET SQL_TRACE = FALSE-- Stop Session Tracing

C:\orcl\product\10.2.0\admin\TUMAY\udump>TKPROF
C:\orcl\product\10.2.0\admin\TUMAY\udump\tumay_ora_6372     test.txt
explain=spatialndx/pass@tumay table=sys.plan_table

TKPROF: Release 10.2.0.1.0 - Production on Sun Oct 18 18:02:35 2009

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Üretilen text dosyasına baktığımızda;

call     count  cpu    elapsed       disk      query    current
rows
------- ------ ------- ---------- ---------- ---------- ---------
--------
Parse        1   0.03       9.67          0        831          0
0
Execute      1   0.00       0.00          0          0          0
0
Fetch        4   0.12       2.72          0         54          2
32
------- ------  ------ ---------- ---------- ---------- -------  ---
-------
total        6   0.15      12.39          0        885          2
32
```

Ek F

```
function Agiparser(dAgiAdi)
        {

          format = 'image/png';
       if(pureCoverage) {
document.getElementById('filterType').disabled = true;
document.getElementById('filter').disabled = true;
document.getElementById('antialiasSelector').disabled = true;
document.getElementById('updateFilterButton').disabled = true;
document.getElementById('resetFilterButton').disabled = true;
document.getElementById('jpeg').selected = true;
format = "image/jpeg";
                }

var bounds = new OpenLayers.Bounds(
         25.0977295, 35.4999238,
         45.766410500000006, 42.422426200000004
          );
var options = {
         controls: [],
         maxExtent: bounds,
         maxResolution: 0.08073703515625,
         projection: "EPSG:4326",
         units: 'degrees'
          };
     map = new OpenLayers.Map('map', options);

 tiled = new OpenLayers.Layer.WMS(

 "topp:havzalar - Tiled", "http://localhost:8989/geoserver/wms",
                     {
                         height: '300',
                         width: '800',
                         layers: 'topp:havzalar',
                         styles: '',
                         srs: 'EPSG:4326',
                         format: format,
                         tiled: 'true',
                         tilesOrigin : "25.0977295,35.4999238"
                     },
                     {buffer: 0}
                 );




 tumAkarsular = new OpenLayers.Layer.WMS(
                     "topp:AKARSU - Tiled",
"http://localhost:8989/geoserver/wms",
                     {
                         height: '300',
                         width: '800',
                         layers: 'topp:AKARSU',
                         styles: '',
```

```
                        srs: 'EPSG:4326',
                        format: format,
                        tiled: 'true',
                        tilesOrigin : "25.0977295,35.4999238"
                    },
                    {buffer: 0}
                );


            var style_green = {
                  strokeColor: "#00FF00",
                  strokeOpacity: 1,
                  strokeWidth: 3,
                  pointRadius: 6,
                  pointerEvents: "visiblePainted"
             };


            tumAkarsular = new OpenLayers.Layer.WMS("Tüm Akarsular",
             "http://localhost:8989/geoserver/wms",
             {'layers': 'topp:AKARSU', transparent: true, format:
'image/gif'},
             {isBaseLayer: false}
                 );


    markers = new OpenLayers.Layer.Markers("Markers");
    layerMarkers = new OpenLayers.Layer.Markers("Markers");
    var size = new OpenLayers.Size(21,25);
    var offset = new OpenLayers.Pixel(-(size.w/2), -size.h);
    var icon = new
OpenLayers.Icon('http://localhost/googlemaps/images/blueMarker.jpg',
size,offset);

var layer_style = OpenLayers.Util.extend({},
OpenLayers.Feature.Vector.style['default']);
layer_style.fillOpacity = 0.2;
layer_style.graphicOpacity = 1;


var request = OpenLayers.Request.GET({url: dAgiAdi,async: false});
var data =request.responseXML;
 placemarks =
data.documentElement.getElementsByTagName("Placemark");
 var vectorLayer = new OpenLayers.Layer.Vector("Simple Geometry",
{style: layer_style});

 for(var i=0; i<placemarks.length; i++){
                var coordinates;
                        coordinates =
placemarks[i].getElementsByTagName("coordinates")[0].childNodes[0].n
odeValue
                    var name =
placemarks[i].getElementsByTagName("name")[0].childNodes[0].nodeValu
e;
```

```
                        for(var chunk=1;
chunk<placemarks[i].getElementsByTagName("coordinates")[0].childNode
s.length;chunk++){

coordinates+=placemarks[i].getElementsByTagName("coordinates")[0].ch
ildNodes[chunk].nodeValue;
                        }
                        coordinates = coordinates.split(" ");
                        for(var j=0; j<coordinates.length;j++){
                              coordinates[j] =
coordinates[j].split(",");
                        }
                        if(coordinates.length == 1){
                              var point = new
GLatLng(parseFloat(coordinates[0][1]),parseFloat(coordinates[0][0]))
;
var desc =null;

var marker =new OpenLayers.Marker(new
OpenLayers.LonLat(parseFloat(coordinates[0][0]),parseFloat(coordinat
es[0][1])),new
OpenLayers.Icon("http://localhost/googlemaps/images/blueMarker.jpg",
new OpenLayers.Size(15,25)));
marker.id  = '<p>'+ name;
marker.x=parseFloat(coordinates[0][0]);
marker.y=parseFloat(coordinates[0][1]);

marker.popupClass = OpenLayers.Popup.FramedCloud;
marker.closeBox = true;

layerMarkers.addMarker(marker);
marker.events.register("click", marker, function(e) {

var lonlat = map.getLonLatFromViewPortPx(e.xy);
 popup = new OpenLayers.Popup("agi",
         new OpenLayers.LonLat(this.x,this.y),
         new OpenLayers.Size(130,110),
         this.id,
         true);
            popup.closeOnMove = true;
            popup.backgroundColor ="lightskyblue";
            map.addPopup(popup);
});
                        }//if(coordinates.length == 1)

                              else
                              {
                    var points = new Array();


var lonLat = new
GLatLng(parseFloat(coordinates[0][1]),parseFloat(coordinates[0][0]))
;
var polygonFeature = new OpenLayers.Feature.Vector(new
OpenLayers.Geometry.Point(parseFloat(coordinates[0][0]),parseFloat(c
oordinates[0][1])),null, style_green);
               vectorLayer.addFeatures([ polygonFeature]);
```

```
var marker =new OpenLayers.Marker(new
OpenLayers.LonLat(parseFloat(coordinates[0][0]),parseFloat(coordinat
es[0][1])),new
OpenLayers.Icon("http://localhost/googlemaps/images/blueMarker.jpg",
new OpenLayers.Size(15,25)));
marker.id  = '<p>'+ name;

marker.x=parseFloat(coordinates[0][0]);
marker.y=parseFloat(coordinates[0][1]);
layerMarkers.addMarker(marker);
marker.events.register("click", marker, function(e) {
    var lonlat = map.getLonLatFromViewPortPx(e.xy);


            popup = new OpenLayers.Popup("agi",
            new OpenLayers.LonLat(this.x,this.y),
            new OpenLayers.Size(130,110),
            this.id,
            true);
            popup.closeOnMove = true;
            popup.backgroundColor ="lightskyblue";
            map.addPopup(popup);
});


  }
        }  //for

          map.addLayers([tiled,tumAkarsular]);

          map.addControl(new OpenLayers.Control.PanZoomBar({
               position: new OpenLayers.Pixel(2, 15)
               }));
          map.addControl(new OpenLayers.Control.Navigation());
          map.addControl(new OpenLayers.Control.Scale($('scale')));
          map.addControl(new
OpenLayers.Control.MousePosition({element: $('location')}));
          map.addLayer(layerMarkers);



         try
  {

           var gphy = new OpenLayers.Layer.Google(
               "Google Physical",
               {type: G_PHYSICAL_MAP}
           );
           var gmap = new OpenLayers.Layer.Google(
               "Google Streets",
               {numZoomLevels: 20}
           );
           var ghyb = new OpenLayers.Layer.Google(
               "Google Hybrid",
               {type: G_HYBRID_MAP, numZoomLevels: 20}
```

```
            );
            var gsat = new OpenLayers.Layer.Google(
                "Google Satellite",
                {type: G_SATELLITE_MAP, numZoomLevels: 20}
            );


            map.addLayers([gphy, gmap, ghyb, gsat]);

            map.setCenter(new GLatLng(39.339797, 33.330864), 4);

            map.addControl(new OpenLayers.Control.LayerSwitcher());
}
catch(err)
  {

  txt+="Click OK to continue.\n\n";
  alert(txt);
  }



 map.setCenter(new GLatLng(39.339797, 33.330864), 4);
                map.zoomToExtent(bounds);

            }
```

Ek G

```
CREATE OR REPLACE PROCEDURE GETLINKIDS (node_id in
number,objid_string out varchar2) is
y sdo_number_array;
x number;
link_ids sdo_number_array;
i NUMBER;

BEGIN
sdo_net_mem.network_manager.read_network('RIVER', 'FALSE');
link_ids:=SDO_NET_MEM.NODE.GET_INCIDENT_LINK_IDS( 'RIVER',node_id);
i:=link_ids(link_ids.last);
for e in
(
select link_id  from river_link where link_id <=
link_ids(link_ids.last)
) loop

if i=1 then
objid_string:=objid_string||e.link_id;
x:=x-1;
else
objid_string:=e.link_id||','||objid_string;
x:=x-1;
end if;

i:=i-1;
end loop;

SDO_NET_MEM.NETWORK_MANAGER.DROP_NETWORK('RIVER');

END GETLINKIDS ;
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : KÜTÜKCÜ, Arzu
Uyruğu : T.C.
Doğum tarihi ve yeri : 16.08.1978 Samsun
Medeni hali : Bekar
Telefon : 0 (312)
Faks : 0 (312)
e-mail : akutukcu@etu.edu.tr

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Karadeniz TeknikÜniversitesi/Bilgisayar Mühendisliği	1999

İş Deneyimi

Yıl	**Yer**	**Görev**
2000-2005	**GenKur.MUBİLDESKOM/Ankara**	**Bilgisayar Müh.**
2005-	**Devlet Su İşleri Genel Müdürlüğü/Ankara**	**Bilgisayar Müh.**

Yabancı Dil

İngilizce

Yayınlar