

**TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**ZAMAN UZAMSAL SENTETİK VERİ SETİ ÜRETİMİ VE
SORGULANMASI**

18011903 – Ramiz Mammadli
18011071 – Elif Yağmur Duran

BİLGİSAYAR PROJESİ

Danışman
Dr. Ögr. Üyesi Mustafa Utku Kalay

Ocak, 2022

TEŞEKKÜR

Bu projenin tamamlanmasında bize yardımcı olan rehber öğretmenimiz sayın Dr. Öğr. Üyesi Mustafa Utku Kalay'a teşekkür ederiz.

Ayrıca rehberlik konusunda yardımlarını sunan sınıf arkadaşlarımıza özel olarak teşekkürlerimizi sunarız.

Ramiz Mammadlı
Elif Yağmur Duran

İÇİNDEKİLER

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
ÖZET	viii
ABSTRACT	ix
1 Giriş	1
2 Ön İnceleme	2
3 Fizibilite	3
3.1 Teknik Fizibilite	3
3.1.1 Yazılım Fizibilitesi	3
3.1.2 Donanım Fizibilitesi	4
3.1.3 İş ve Zaman Fizibilitesi	4
3.1.4 Yasal Fizibilite	4
3.1.5 Ekonomik Fizibilite	5
4 Sistem Analizi	6
5 Sistem Tasarımı	7
5.1 Ön Hazırlık	7
5.2 QGIS Ortamı ile Bağlantı Kurulması ve Diğer Shapefile'ların Eklenmesi	8
6 Uygulama	10
6.1 QGIS'te Sentetik Veri Üretimi	10
6.1.1 CRS formatının ayarlanması	10
6.1.2 Rota Üretimi	11
6.1.3 Filtreleme İşlemleri	11
6.1.4 Nokta üretimi	13
6.1.5 Üretilmiş noktalara zaman etiketi atanması	13
6.2 PostGIS'te 'geom' Sütununa Zaman Boyutunun Eklenmesi	16

7 Deneysel Sonuçlar	21
7.1 Indexleme İşleminin Açıklaması ve Kullanılma Sebepleri	21
7.2 Kullanılan Indexler	22
7.2.1 GIST	22
7.2.2 SPGIST	22
7.2.3 BRIN	22
8 Performans Analizleri ve Yapılan Çıkarımlar	24
8.1 Analizler Yapılırken Dikkat Edilen Faktörler	24
8.2 Analizi Yapılan Tablo	24
8.3 Explain Analyze Sorgusu İle Çıkarımlar	25
8.3.1 Indexsiz	25
8.3.2 GIST	26
8.3.3 SPGIST	27
8.3.4 BRIN	27
8.4 Grafikler ve Çıkarımlar	27
9 Sonuç	30
Referanslar	31
Özgeçmiş	32

KISALTMA LİSTESİ

OSM	Open Street Map
SQL	Structured Query Language
GIS	Geographic Information System
CRS	Coordinate Reference System
.shp	Shapefile
QGIS	Quantum Geographic Information System
API	Application Programming Interface

ŞEKİL LİSTESİ

Şekil 3.1 Gantt Diyagramı	4
Şekil 5.1 PostGIS uzantısının bağlanması	7
Şekil 5.2 Geometri tablosunun oluşturulması	8
Şekil 5.3 QGIS'e, indirilen .shp uzantılı dosyaların aktarılması	9
Şekil 5.4 Ham verinin QGIS ortamında görüntülenmesi	9
Şekil 6.1 Projenin CRS formatı değiştirildikten sonraki görüntüsü	11
Şekil 6.2 Rastgele seçilmiş sokaklar	12
Şekil 6.3 80 metre ve üstü, rasgele seçilmiş sokakların bir kısmı	12
Şekil 6.4 Önceden rasgele seçilmiş rotalar üzerinde, 40 metreden bir üretilmiş noktalar	13
Şekil 6.5 Noktalara timestamp atanması için yazılan expression	14
Şekil 6.6 Noktanın bulunduğu çizgi üzerinde açısal değerinin bulunması .	15
Şekil 6.7 Zaman etiketlerinin dönüşlere göre güncellenmesi	15
Şekil 6.8 Noktaların özelliklerini gösteren veriseti	16
Şekil 6.9 LAST_POINTS tablosundaki değerlerin yerleşme şekli	17
Şekil 6.10 LAST_POINTS tablosunda ST_X(geom) ve ST_Y(geom) değerlerinin gösterimi	18
Şekil 6.11 SELECT EXTRACT fonksiyonunun test edilmesi	18
Şekil 6.12 ST_MakePointM fonksiyonunun test edilmesi	19
Şekil 6.13 AddGeometryColumn fonksiyonunun çalıştırılması	19
Şekil 6.14 insert into ile yeni tabloya kayıtların eklenmesi	20
Şekil 6.15 Yeni tablonun son hali	20
Şekil 7.1 Bounding Boxların yerleşimi	21
Şekil 7.2 Create Index Using GIST sorgusu	22
Şekil 7.3 Create Index Using SPGIST sorgusu	22
Şekil 7.4 Create Index Using BRIN sorgusu	23
Şekil 8.1 points_3dgeom tablosunun geom değerlerine göre sıralı sorgusunun sonucu	25
Şekil 8.2 ST_3DDWithin ile indexsiz EXPLAIN ANALYZE sorgusu	26
Şekil 8.3 ST_3DDWithin ile GIST indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu	26

Şekil 8.4 ST_3DDWithin ile SPGIST indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu	27
Şekil 8.5 ST_3DDWithin ile BRIN indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu	28
Şekil 8.6 EXPLAIN ANALYZE sorgularında total runtimeların karşılaştırılması	28
Şekil 8.7 EXPLAIN ANALYZE sorgularında planning timeların karşılaştırılması	29
Şekil 8.8 EXPLAIN ANALYZE sorgularında execution timeların karşılaştırılması	29

ÖZET

ZAMAN UZAMSAL SENTETİK VERİ SETİ ÜRETİMİ VE SORGULANMASI

Ramiz Mammadli

Elif Yağmur Duran

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Dr. Ögr. Üyesi Mustafa Utku Kalay

İstanbul günümüzde dünyanın en ünlü ve kalabalık metropol şehirlerinden biri olarak bilinmektedir. 2500 yıllık tarihi boyunca birçok medeniyete ev sahipliği yapmış olan şehrimiz, yıllar içerisinde farklı geçmişlerden birçok insanı barındırmayı başarmıştır. Şehirden geçen her yeni insan grubu ile yeni yollar, sokaklar ve mahalleler inşa edilmiştir ve imar asla durmamıştır. Dolayısıyla da şehrin haritası 2000 yıldır değişmektedir. Şüphesiz, tarihsel olarak böylesine karmaşık bir haritadan uzam-zamansal veri üretmek ve performans analizi gibi işlemler uygulamak da aynı derecede zor olacaktır. Bu projede de tam olarak bunu yapmayı hedefledik. Mentörümüz Dr. Ögr. Üyesi Utku Kalay'ın yardımıyla QGIS üzerinde bir uzam-zamansal veri tabanı oluşturduk ve bunu PostGIS aracı yardımıyla PostgreSQL ortamına yüklemeyi başardık. Ardından, işleri daha verimli hale getirmek için verilerimize belirli optimizasyon prosedürleri uyguladık. Burada yazdığımız scriptler ile sokakların belirli koşullarını göz önünde bulundurarak oluşturulan rotalara bir zaman faktörü eklemeyi başardık. Son olarak sentetik verilerimizi PostgreSQL ortamına geri çekerek 3 boyutlu zaman uzamsal veri haline getirdik, çeşitli indeksleme yöntemleri uyguladık ve performans analizleri yaptık. Yaptığımız analizle, birkaç indeksleme tekniklerinin farklılıklarını, artıları ve eksileri, bellek kullanım karakteristikleri, PostgreSQL'in indeksleri kullanma konusunda nasıl çalıştığı gibi birçok farklı faktörü gözlemledik, proje sonunda da bulgularımızı görselleştirdik.

Anahtar Kelimeler: Uzam-zamansal veri seti, yol analizi, PostGIS, QGIS.

ABSTRACT

SPATIO - TEMPORAL SYNTHETIC DATASET GENERATION AND QUERYING

Ramiz Mammadli

Elif Yağmur Duran

Department of Computer Engineering

Computer Project

Advisor: Asst. Prof. Mustafa Utku Kalay

Istanbul is known to be one of the most famous and crowded metropolitan cities in the world. Our city, which has been a home to many civilizations throughout its 2500-year history, has managed to withstand housing many people of different backgrounds throughout the years. With every new group of people passing through the city, new roads, streets and neighborhoods were built, and the reconstruction never stopped. For this reason in particular, the map has been evolving for the past 2000 years. Surely, the analysis of such a historically complex map would be just as difficult. In this project, we aimed do exactly that. With the help of our mentor, Associate Professor Utku Kalay, we generated a spatio-temporal database on QGIS and managed to upload that into PostgreSQL environment, with the aid of PostGIS tool. Then we applied certain optimization procedures to our data, in order to make things more efficient. With the scripts we have written here, we managed to add a time factor to the generated routes, considering the certain conditions of the streets. Lastly, we pulled our synthetic data back to PostgreSQL environment to make it 3-dimensional spatio-temporal data, apply various indexing methods and conduct some performance analysis. With the analysis we performed, we observed on many different factors, such as, the differences and the pros and cons of different indexing techniques, how they place themselves into the memory, and how PostgreSQL operates on using the indexes, then we made visual representations for our findings.

Keywords: Spatio-temporal data set, road analysis, PostGIS, QGIS

1

Giriş

İstanbul dünyanın en ünlü ve kalabalık metropol şehirlerinden biridir, 2500 yıllık tarihiyle yıllara karşılık ayakta durmuş bir şehirdir. Dünyanın eski şehirlerinin yol haritalarının karmaşık ve dolambaçlı olduğu bilinen bir gerektir, İstanbul'da bunlara bir örnektir. Biz projemizde bu karmaşık haritanın analiziyle uğraşarak uzamsal veritabanları hakkında kendimizi geliştirmeyi amaçladık.

Peki uzamsal veritabanı nedir? Kısaca geometrik özelliklere sahip olan verileri saklayan veritabanı olarak tanımlanabilir. Günümüzde bilgisayar bilimi, kullanılan araçlar ve diller konusundaki popüler konseptlerin çok hızlı değiştiği bir alan olarak bilinmektedir. Özellikle son yıllarda verinin analizi, saklanması, yönetilmesi ile ilgili meslekler yükselişe geçmiş bulunmaktadır. Ancak yine de günümüz yazılımcıları arasında yapılan bir ankette kendilerine tanıdık gelen araçları kullanmaya daha eğilimli oldukları saptanmıştır. Yani popüler konseptler çabuk yükseliyor olsa bile, çoğu meslektaşımız, hala istenen sonuçlara ulaşmak için keşfedilen, var olan yöntemlerin daha gelişmiş halini kullanıcıya sunan yeni araçları tercih etmemektedir. Tam olarak da bu sebeple de uzamsal SQL'in gizli potansiyeli henüz çoğu yazılımcı tarafından keşfedilmemiş durumdadır.

Biz bu projemizde uzam-zamansal veritabanları hakkında araştırmalarda bulunduk ve bu alanın sunduğu imkanlar hakkında bilgi topladık. Daha sonra öğrendiklerimizi QGIS, PostGIS ve PostgreSQL gibi ortamlarda test ettik. Projemizi implemente ederken açık kaynak kodlu ve ücretsiz veritabanı yönetim araçlarından faydalandık.

2 Ön İnceleme

Geçtiğimiz birkaç yıl içerisinde uzamsal SQL'in sunduğu imkanlar sayesinde, geleneksel veri tabanı sorgulama yöntemleri ile çözemediğimiz problemleri çözebilen birçok proje ortaya konulmuştur. Örnek vermek gerekirse, Stockon Üniversitesi öğrencileri arasında yapılan bir yarışmada, potansiyel yaban mersini çiftlik konumları analizinden [1] çevresel ırkçılığın etkilerinin coğrafi dağılımına [2] kadar pek çok değişik konuya ilgili projeler çıkarılmıştır. Buna ek olarak, beyinin hafızayı nasıl formüle ettiği ile ilgili bir araştırmada uzam-zamansal veri setleri üretilmiş ve analiz edilmiştir [3]. Bu araştırmalar sayesinde uzamsal SQL alanının önemi bir kez daha göz önüne konulmuş oldu.

Projede İstanbul karayolları haritasından rastgele olarak sokaklar seçtik ve yeni bir veri seti oluşturduk. Bu oluşturulan veri seti üzerinde belirli işlemler uyguladık ve geliştirdiğimiz modele göre sentetik veri seti elde etmiş olduk. Sonrasında, bu veriyi 3 boyutlu uzam-zamansal hale getirerek 3 farklı indeksleme yöntemleri uyguladık. Sonuç olarak, indekslenmiş ve indekssiz veri seti arasındaki karşılaştırmayı göstermek için performans analizi yaptık ve bunu grafikler sayesinde görselleştirdik. Bu sayede, indeksleme yönteminin büyük veriler için iyi olup olmadığı, iyi ise her yöntem her veri seti tipi için iyi mi sonucunu elde etmiş olduk.

3

Fizibilite

Bu bölümde projenin teknik, iş ve zaman, yasal ve ekonomik açıdan fizibilite çalışmaları gerçekleştirılmıştır.

3.1 Teknik Fizibilite

Bu bölümde yazılım ve donanım fizibilitesi ayrıntılı olarak gerçekleştirılmıştır.

3.1.1 Yazılım Fizibilitesi

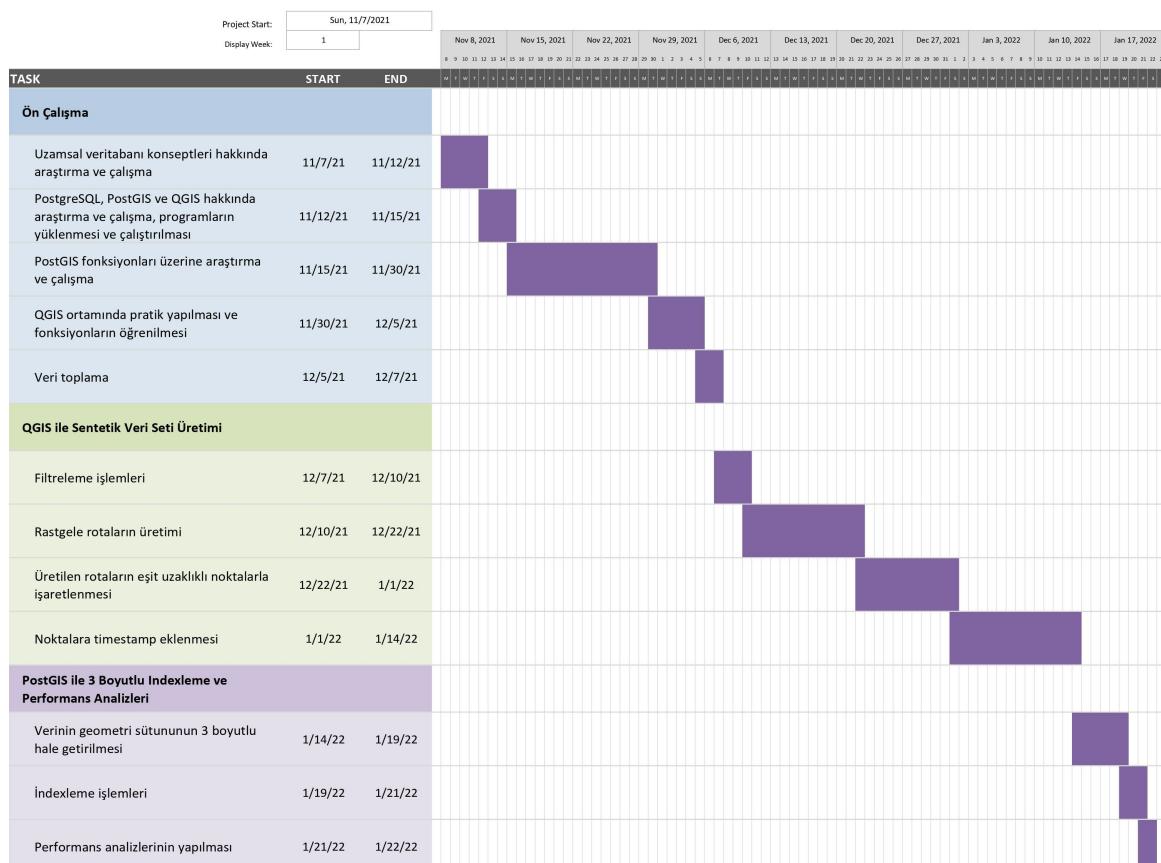
- **QGIS:** QGIS, coğrafi verilerin görüntülenmesini, düzenlemesini ve analizini destekleyen ücretsiz ve açık kaynaklı 'cross platform' coğrafi bilgi sistemi (GIS) uygulamasıdır. Bu uygulama sayesinde farklı formatlarda indirilen veri setlerini inceleme, katmanlara göre analiz yapma, ham veriye göre sentetik veriseti üretme işlemleri rahatlıkla yapılır.
- **PostgreSQL:** Postgres olarak da bilinen PostgreSQL, genişletilebilirliği ve SQL uyumluluğunu vurgulayan ücretsiz ve açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir.
- **POSTGIS:** POSTGIS, coğrafi nesneler için destek ekleyen açık kaynaklı bir PostgreSQL veritabanı uzantısıdır. Birçok geometri türlerini, mekansal ve zaman-uzaysal operatörleri, uzamsal ve jeo-uzamsal küme işlemlerini içerisinde barındırır ve zaman-uzamsal veriseti üretiminde en çok tercih edilen teknolojilerden bir tanesidir.
- **Python:** Python, yorumlanmış üst düzey genel amaçlı bir programlama dilidir. Tasarım felsefesi, önemli girinti kullanımıyla kod okunabilirliğini vurgular. Dil yapıları ve nesne yönelimli yaklaşımı, programcının küçük ve büyük ölçekli projeler için açık, kodlar yazmasına yardımcı olmayı amaçlar.

- Matplotlib:** Matplotlib, Python için bir cross platform, veri görselleştirme ve grafik çizim kütüphanesidir. Bu nedenle, birçok kullanılan açık kaynak olmayan uygulamalara uygun bir alternatif sunar. Projede Matplotlib, performans analizi sırasında kaydedilen sonuçları görselleştirmek için kullanıldı.

3.1.2 Donanım Fizibilitesi

Projenin geliştirilmesinde 2 bilgisayar kullanıldı. Birinci bilgisayar, Intel Core i7-1165G7 CPU, 16 GB RAM, Nvidia MX450 GPU ve 1 TB SSD, ikinci bilgisayar ise Intel Core i7-6700HQ CPU, 16 GB RAM, Nvidia GTX960M GPU ve 512 GB SSD'ye sahiptir. Bu özellikler sistemin geliştirilmesi ve implementasyonu için yeterli bir donanım sağlamaktadır.

3.1.3 İş ve Zaman Fizibilitesi



Şekil 3.1 Gantt Diyagramı

3.1.4 Yasal Fizibilite

Kullandığımız yazılımlar ve araçlar açık kaynak kodludur. Bu yazılımlar ve araçların ticari kullanım dışında herhangi bir kısıtlaması yoktur. Projemiz ticari

amaçlı olmadığından kullandığımız tüm yazılım ve araçlar yasal olarak sorun teşkil etmemektedir.

3.1.5 Ekonomik Fizibilite

Kullandığımız yazılımlar ve araçlar ücretsizdir. Geliştirme yaptığımız cihazlar kişisel cihazlarımız olduğundan altyapı konusunda da mali bir yükümlülüğümüz bulunmamaktadır.

4 Sistem Analizi

Projenin ilk kısmında QGIS üzerinden İstanbul haritasının karayolları verisi üzerinden rastgele yerleştirilmiş noktalar arasında rotalar üretilmiştir. İleride performans analizleri yapılırken fark görülebilmesi için verinin büyük olması amaçlanmıştır. Bu rotalar üzerinde eşit uzaklıkta noktalar yerleştirilip, elde edilen noktalar tablosuna zaman boyutu eklenmiştir. Eklenen bu ‘timestamp’ sütununda ilk adımda her noktanın arasında eşit zaman uzaklığı vardır. Projedeki gerçekçiliği artırabilmek için noktaların bulunduğu yol üzerinde dönüş açıları hesaplanarak, belli bir açıdan büyük değişimlere rastlandığında zaman bilgisine gecikme eklenir. Rotalar ve noktalar, İstanbul yolları ile beraber görsel bir sunum yapılabilmesi için İstanbul'un binalar, su yolları, ray hatları gibi diğer uzamsal veri tabloları ile birlikte gösterilir.

İkinci kısmında ise üzerinde işlemler yapılmış veri üzerinden yeni üretilen rotalar ve noktalar tabloları PostGIS'e geri gönderilecek ve üzerinde 3 farklı index tipinden farklı veri boyutları üzerinde performans analizleri yapılacaktır. Bu indeks tiplerinin aralarındaki farklar faydalari ve zararları üzerinden açıklanacak, PostgreSQL'in bunları hangi şartlar altında nasıl kullanmayı tercih ettiğinden bahsedilecektir. Indexlerin veriye yerleşme ve sorgulara cevap getirme süreleri arasında yapılan karşılaştırmalardan alınan sonuçlar da bu kısımda görselleştirilmiştir.

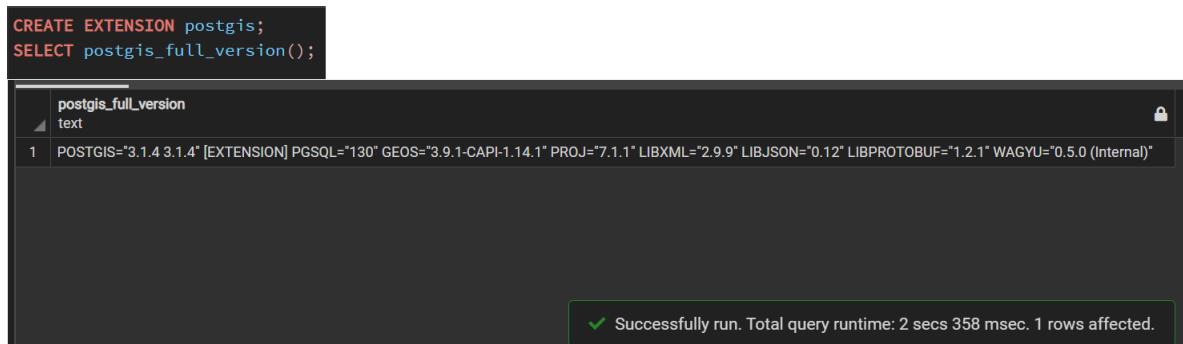
5 Sistem Tasarımı

5.1 Ön Hazırlık

Bu başlık altında projenin başında gerekli olan hazırlıklar kısaca anlatılıp incelenecaktır.

Projede kullanılan haritanın ilk başta 'OpenStreetMap' üzerinden çekilmesi planlandı. Ancak, çekeceğimiz harita verisi boyut olarak 'OpenStreetMap'in, kendi websitesinin belirlediği üst sınırda daha büyük olduğu için bu yöntemden vazgeçildi ve bu tip senaryolar için alternatif olarak kullanılan OSM tabanlı BBBike API'sı kullanıldı. Yukarıda belirtilen her iki kaynağın ücretsiz hizmet verdığının de belirtilmesi gereklidir. Çekilen veri içerisinde .shp formatında 'buildings', 'landuse', 'natural', 'places', 'points', 'railways', 'waterways' ve 'roads' olmak üzere 8 farklı tablo vardır. İşlemler 'roads' tablosu üzerinde yapıldı ve diğer shp dosyaları QGIS ortamında görselliği artırmak için kullanıldı.

İlk adımda, PgAdmin4 ortamında, PostgreSQL kullanılarak boş bir veritabanı açıldı ve PostGIS uzantısı bu veritabanına bağlandı. PostGIS veritabanına bağlandıktan sonra otomatik olarak bir "spatial_ref_sys" tablosu oluşturur (Şekil 5.1).



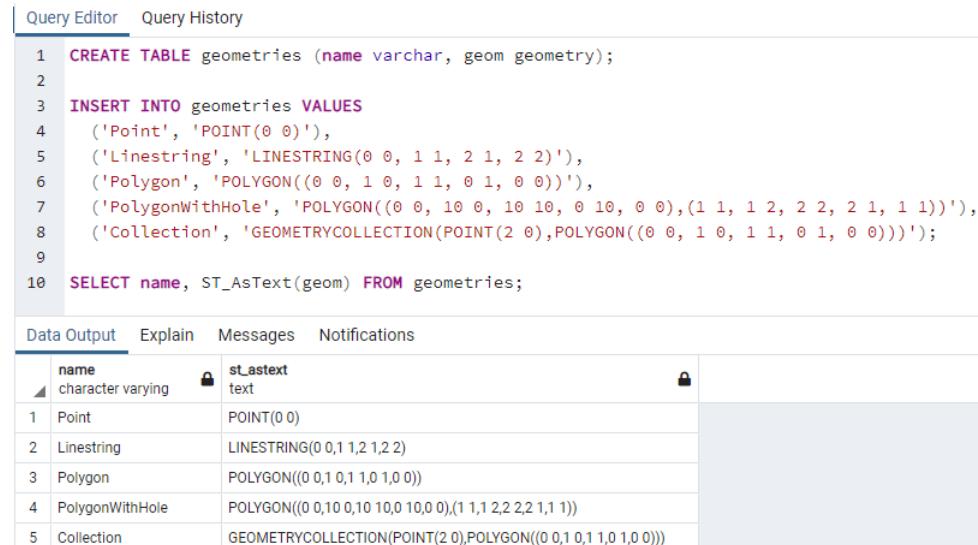
```
CREATE EXTENSION postgis;
SELECT postgis_full_version();
```

postgis_full_version	text
1	POSTGIS="3.1.4 3.1.4" [EXTENSION] PGSQL="130" GEOS="3.9.1-CAPI-1.14.1" PROJ="7.1.1" LIBXML="2.9.9" LIBJSON="0.12" LIBPROTOBUF="1.2.1" WAGYU="0.5.0 (Internal)"

Successfully run. Total query runtime: 2 secs 358 msec. 1 rows affected.

Şekil 5.1 PostGIS uzantısının bağlanması

Ardından, geometri çeşitlerini temsil etmesi için bir "geometries" tablosu oluşturuldu. Bu tablo sayesinde, projenin ilerileyen aşamalarında kullanılabilecek fonksiyonlara ortam hazırlanmış oldu (Şekil 5.2).



```

Query Editor Query History
1 CREATE TABLE geometries (name varchar, geom geometry);
2
3 INSERT INTO geometries VALUES
4     ('Point', 'POINT(0 0)'),
5     ('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
6     ('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
7     ('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))'),
8     ('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');
9
10 SELECT name, ST_AsText(geom) FROM geometries;

```

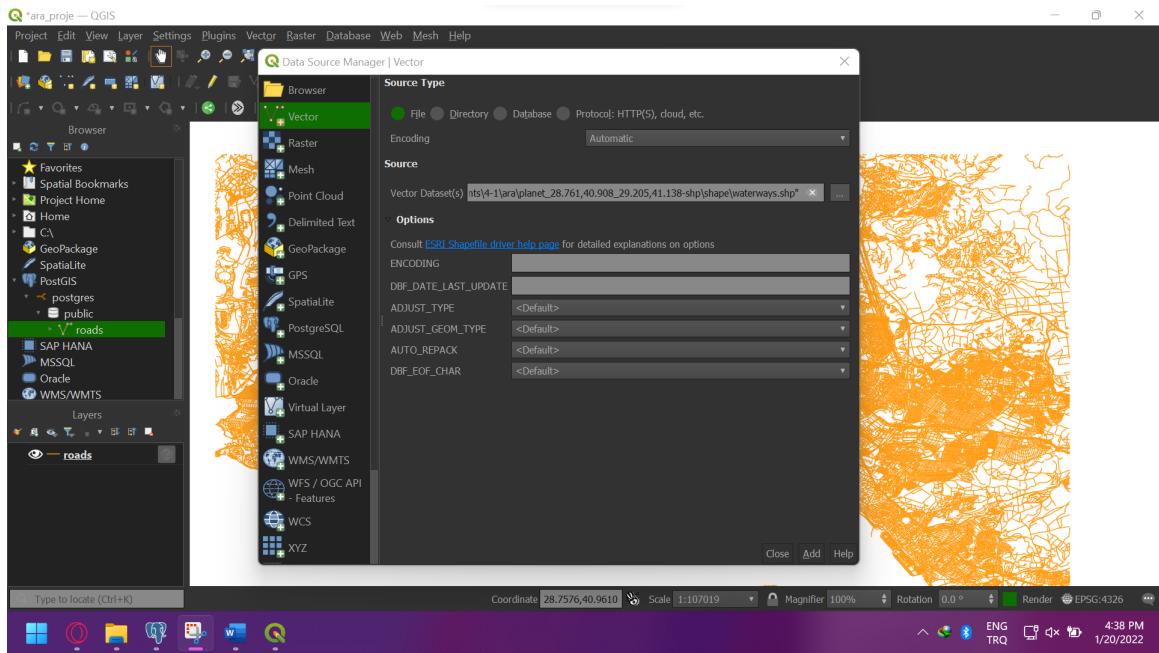
Data Output		Explain	Messages	Notifications
	name character varying	st_astext text		
1	Point	POINT(0 0)		
2	Linestring	LINESTRING(0 0,1 1,2 1,2 2)		
3	Polygon	POLYGON((0 0,1 0,1 1,0 1,0 0))		
4	PolygonWithHole	POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 2,2 1,1 1))		
5	Collection	GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0)))		

Şekil 5.2 Geometri tablosunun oluşturulması

5.2 QGIS Ortamı ile Bağlantı Kurulması ve Diğer Shapefile'ların Eklenmesi

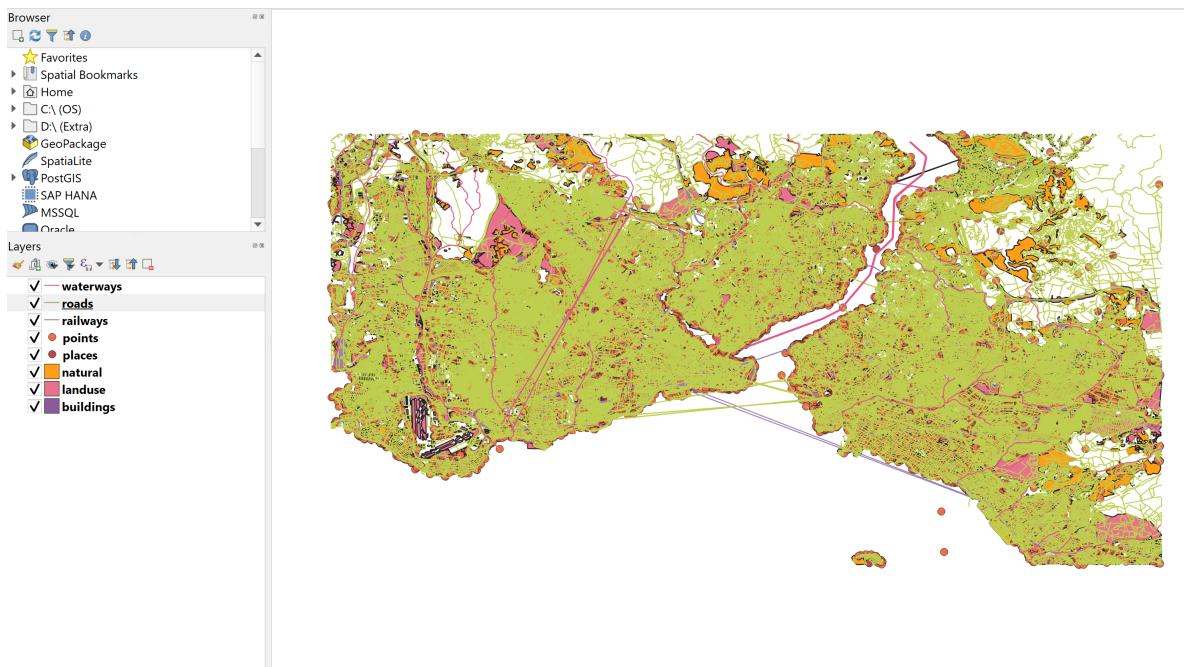
Verinin işlenebilmesi için PostGIS ile QGIS arasında bağlantı kuruldu. Bunun için QGIS'te yeni bir proje açılıp PostGIS'e kendi kullanıcı ismimiz üzerinden bağlantı sağlandı. İlk başta, önceden bulunan ham veri QGIS ortamına eklenecek ve sentetik veri üretildikten sonra PostGIS ortamına bağlanacak.

Çekilen ham İstanbul verisinin içerisinde, öncesinde de belirtildiği gibi, veriler 'vector layer' olarak tutuluyor. BU yüzden, QGIS ortamına bu Shapefile'ların eklenmesi için 'Vector Bundle' kullanılması gereklidir.(Şekil 5.3).



Şekil 5.3 QGIS'e, indirilen .shp uzantılı dosyaların aktarılması

Görsellik sağlamaası için OSM'den çekilen verinin diğer tabloları da buraya eklendi. Haritanın QGIS'te, henüz işlenmemiş hali Şekil 5.4de gösterildiği gibidir.



Şekil 5.4 Ham verinin QGIS ortamında görüntülenmesi

6

Uygulama

Bu kısımda, sentetik verinin nasıl üretildiği, indekslendiği gibi önemli noktalar belirtilecektir.

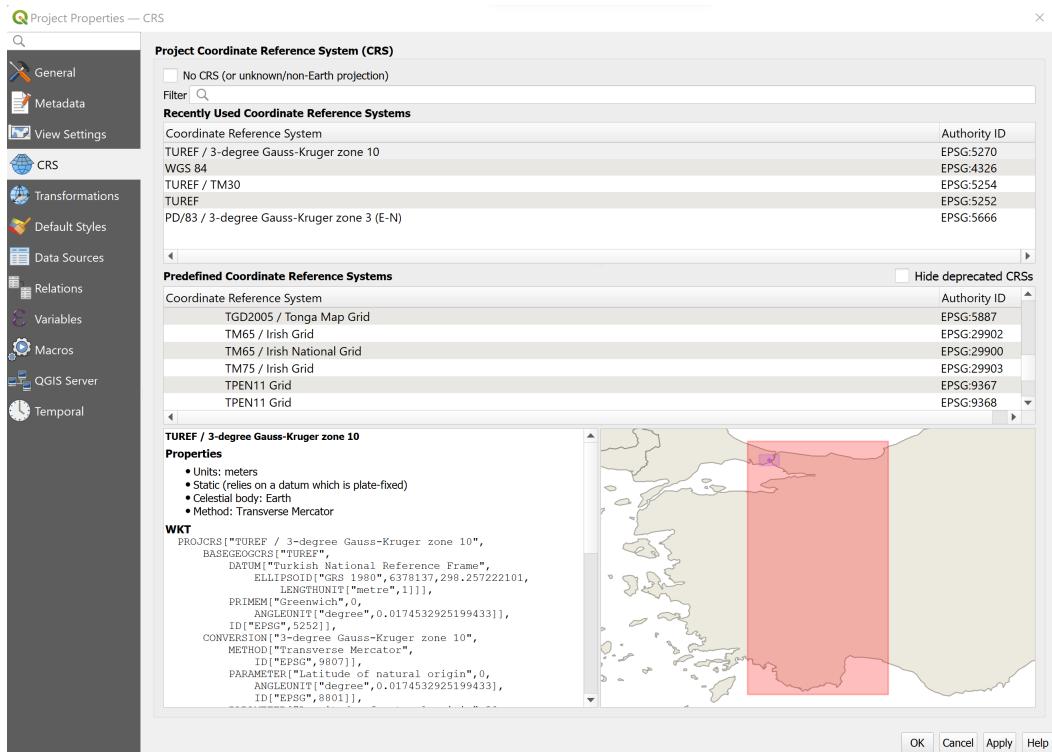
6.1 QGIS'te Sentetik Veri Üretimi

6.1.1 CRS formatının ayarlanması

Şimdiye kadar, İstanbul haritası verisinin internetten nasıl çekilip QGIS ortamına eklendiği konuşuldu. Bu başlık altında, uzam-zamansal sentetik verinin üretimi adım adım anlatılacaktır.

İlk başta, indirilen Shapefile dosyaları içerisindeki 'roads' katmanını incelediğinde zaman CRS formatının WGS 84'e sabitlendiğini söylebilir. QGIS yüklenen her projede otomatik olarak 'Global Projection Specification' olan WGS 84 modunda açmak için ayarlanmıştır [4]. Yapılan proje kıtasal ya da daha büyük bir harita üzerinde işleniyor olsaydı, CRS farkı görmezden gelmek mümkün idi.

Fakat, bu projenin kapsamında İstanbul haritası gibi daha küçük bir alanda çalışıldığı için, projeksiyon farkının göz önünde bulundurulması sonuçların daha az hata payıyla elde edilmesine zemin oluşturacaktır. Buna ek olarak, WGS 84 CRS formatında projeyi çalıştırılırsa, katmanlar üzerinde ölçü birimi derece olacaktır, bu da projenin ilerleyen adımlarında belirli zorluklar oluşturabilir. Bu kapsamında, projenin CRS formatı 'TUREF / 3-degree Gauss-Kruger zone 10'a değiştirilmiştir. İlaveten, önceden eklenen her katmanın da belirtilen CRS formatına getirilmesi gerekmektedir. Şekil 6.1de de görüldüğü gibi, hem ölçü birimi metre olarak sabitlenmiştir hem de harita üzerinde İstanbul'un içerisindeki kesitin alınmasına dikkat edilmiştir. Bu sayede, ölçümler daha net yapılacaktır.



Şekil 6.1 Projenin CRS formatı değiştirildikten sonraki görüntüsü

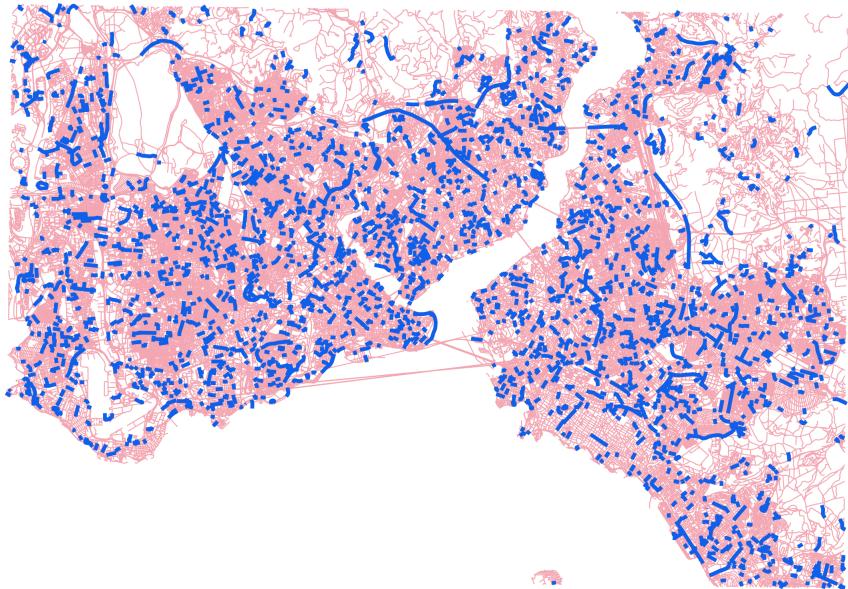
6.1.2 Rota Üretimi

Projenin formatı ayarlandıktan sonra, veri üretme işlemleri başlatıldı. Rota üretimi senaryosu olarak İstanbulda bulunan 88 binden fazla sokaktan ilk olarak 2500 tanesinin seçilmesi ve ayrı bir katman olarak kaydedilmesi kararlaştırıldı. Bunun uygulanması için QGIS'te, 'Processing Toolbox'ta yer alan 'Random Extract' algoritması kullanıldı. Bu algoritma bir vektör katmanı alır ve girdi katmanındaki özelliklerin yalnızca bir alt kümesini içeren yeni bir katman oluşturur. Alt küme, alt kümedeki toplam özellik sayısını tanımlamak için bir yüzde veya sayı değeri kullanılarak rasgele tanımlanır [5]. Bu kurallar esasında, önceden belirtildiği gibi 2500 tane sokak rasgele seçilmiş yeni bir katman olarak tanımlandı (Şekil 6.2).

6.1.3 Filtreleme İşlemleri

Rasgele üretilen rotaların devamında, sentetik veriseti de dahil olmak üzere, diğer tüm katmanlarda hiç kullanılmayacak, ve/veya değerleri NULL olan kolonlar bulunuyordu. Bu kolonlar, 'Attributes Table' penceresinden uygulanan işlemle silindi.

Ek olarak, tahmin edilebileceği üzere, İstanbulda çok kısa (80 metrenin altında) sokaklar da bulunuyor. Bunlar üzerinde işlem yapmak projenin amacına hizmet etmeyecektir. O yüzden, küçük bir filtreleme ile uzunluğu 80 metrenin altında olan tüm sokaklar üretilen verisetinden temizlendi.



Şekil 6.2 Rastgele seçilmiş sokaklar.

Sonuç olarak, Şekil 6.3de gösterilen sentetik bir veriseti elde edildi. Belirtilmesi gereklidir ki, Bazı seçilmiş sokak isimlerinin NULL değerde olması projenin sonuçlarına etki etmemektedir.

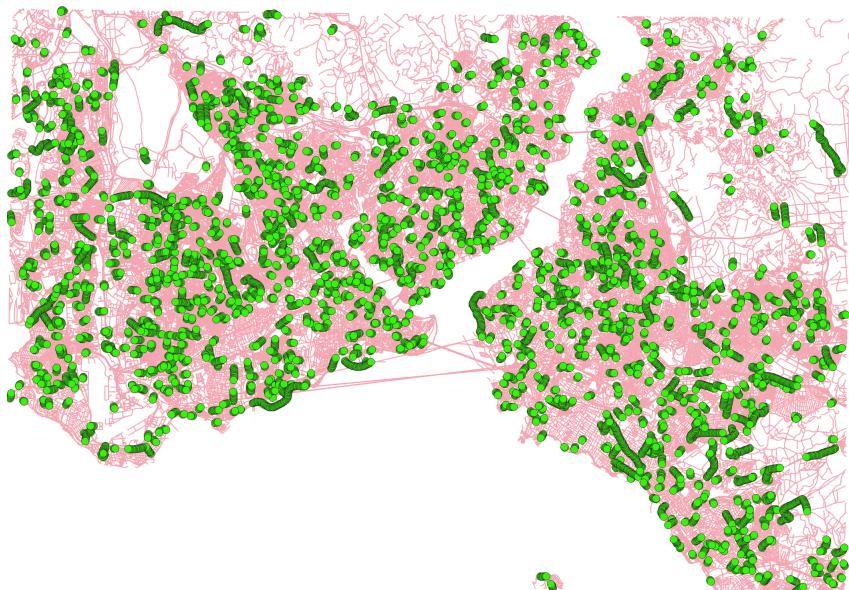
— Features Total: 1432, Filtered: 1432, Selected: 0

	osm_id	name	type
289	88498282	Yan Sokağı	residential
290	89169548	Erman Sokağı	residential
291	89375430	NULL	tertiary
292	89375456	85/7. Sokak	residential
293	90929400	87. Sokak	residential
294	90929435	Beşkardeşler 3. Sokak	residential
295	91140255	8. Bağ Sokağı	residential
296	91227535	Haseki Caddesi	tertiary
297	91233353	Vezir Odaları Sokağı	residential
298	91310354	NULL	footway
299	91353644	Karabulut Sokağı	residential
300	91353655	Türkmen Sokağı	residential
301	91353668	Kuru Çınar Sokağı	residential
302	91387265	Başhoca Sokağı	residential
303	91387287	Hattat Nafız Caddesi	residential
304	91387316	Kıztaşı Caddesi	residential
305	91397283	Cibali Şerefiye Sokağı	residential
306	91397292	Şair Bakı Sokağı	residential

Şekil 6.3 80 metre ve üstü, rastgele seçilmiş sokakların bir kısmı

6.1.4 Nokta üretimi

Her üretilmiş rotanın üzerinde bir aracın harekete hazır olduğunu düşünelim. Eğer, rotaların hepsinin başından sonuna, her 40 metreden bir nokta üretilirse, bu noktalar, o sokak üzerinde hareket eden aracın gittiği yolu temsil eder. Bu noktaları üretmek için Processing Toolbox'ta bulunan 'Points along geometry' algoritmasını kullandık. Bu algoritma, bir girdi vektör katmanının çizgileri boyunca dağıtılan noktalarla bir nokta katmanı oluşturur. Çizgi boyunca olan noktalar arasındaki mesafe parametre olarak tanımlanır. Tahmin edileceği üzere, noktalar önceden üretilen rotalar üzerinde, 40 metreden bir oluşturulmuştur. Sonuç olarak, Şekil 6.4teki gibi bir görüntü elde edilmiştir.



Şekil 6.4 Önceden rasgele seçilmiş rotalar üzerinde, 40 metreden bir üretilmiş noktalar

6.1.5 Üretilmiş noktalara zaman etiketi atanması

Tüm bu işlemlere ek olarak, bütün araçların belirli bir hızla gittiğini varsayıyalım. Hızlarını 20 m/s , yani 72 km/saat olduğunu düşünürsek eğer, araçlar bir noktadan sonrakine iki saniyeye varmış olurlar. Bu mantıkla yola çıktıığında, aracın o noktaya kaç saniyede geldiğini gösteren bir zaman etiketi her noktaya atanabilir.

Fakat, zaman etiketi üretmeden önce, proje katmanlarının shapefile formatından çıkarılması gereklidir. Çünkü, Shapefile formatında yeni bir kolon eklemeye çalışılırsa, veritiplerinin kısıtlı sayıda olduğu farkedilir. Örnek olarak, zaman etiketi eklemek için veri tipinin 'Time' veya 'DateTime' olması gereklidir. Lakin, Shapefile formatında olan katmanların zamanla ilgili sadece 'Date' veri tipi vardır, ve zaman etiketinin atanmasında doğru sonuçlar döndürülmemesine engel olur. Çünkü, üretilecek olan

zaman etiketi saniye hassaslığında olmalıdır, 'Date' veri tipinde ise bu mümkün değildir. Bu yüzden katmanların 'Geopackage' formatına ayarlanması zorunludur. Şöyle ki, Geopackage formatlı katmanlarda 'DateTime' veri tipi vardır ve zaman etiketi atanmasında elverişli ortam hazırlar.

Belirtilen şartlarda bir zaman etiketi (timestamp) eklemek için önceden üretilmiş noktaların 'Attributes Table'ında bulunan 'Field Calculator' sayesinde oluşturulabilir. 'Field Calculator' her katmanın verisini gösteren tabloda bulunan, sentetik veri üretmeye en çok yarayan, içerisinde kod yazılabilen QGIS özelliklerinden birtanesidir. Expression alanına, SQL'e benzeyen, ama kendine has fonksiyonlarını ve syntaxını kullanarak algoritma geliştirilebilir. Bu özellikten faydalananarak, kod geliştirilmiş ve zaman etiketi noktalara atanmıştır. Şekil 6.5te bu amaçla yazılan expression gösterilmiştir. 'Distance' değişkeni noktanın başlangıçtan ne kadar uzakta olduğunu temsil ediyor. Hız ise, 20 m/saniye, yani 72 km/saat olarak kabul edilmiştir. Distance sıfır olduğu anda araç yolculuğa başlamıştır demek oluyor, ve zaman belirlenen tarih ve saatten başlatılıyor. Devamında, her sonraki noktaya varıldığında, noktanın distance değeri önceden belirlenen hız'a bölünüyor ve başlangıç zamanın üzerine ekleniyor. Böylelikle zaman etiketi elde edilmiş oluyor.

```
--ADDING TIMESTAMP TO POINTS

with_variable('speed', 20, -- setting the speed variable
    with_variable('start_time', make_datetime(2022,1,25,20, 0 ,0), -- setting a start time variable
        CASE
            WHEN "distance" = 0
            THEN @start_time -- input the start time at zero distance
            ELSE @start_time + to_interval(to_string("distance" / @speed) || ' seconds') -- adding
                the interval (distance/speed) to the start time and calculating the remaining rows
            END
        )
    )
```

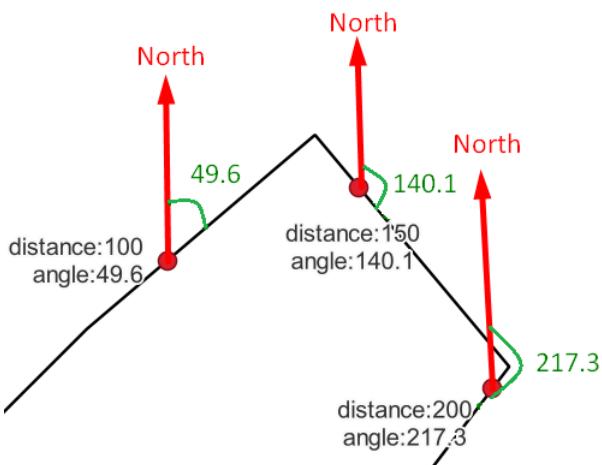
Şekil 6.5 Noktalara timestamp atanması için yazılan expression

6.1.5.1 Yoldaki dönüşlerde aracın hızının azalması

Şimdiye kadar rasgele seçilmiş rotalar üzerinde belirli aralıklarla noktalar üretildi ve o noktalara zaman etiketi atandı. Lakin, atanan zaman etiketleri, yolun koşullarına bakılmaksızın, her belirli aralığı belirli hızla gidiyor. Başka bir deyişle, araçlar her zaman yollarda sabit hızla ilerlemiyorlar. Mesela, eğer yolda belirli bir düzeyde dönüş varsa, aracın hızının azalması gereklidir.

Bu kapsamda, biz de projemizde dönüşlü olan yollarda aracın hızını belirli bir düzeyde azalttık. Yolun dönüşlü olup olmadığını anlamak için üretilen noktalar, üzerinde bulundukları 'linestring'in o andaki azimut (çizginin Kuzey istikametine göre) açısını 'Attributes Table'da tutuyorlar (Şekil 6.6). Bu bilgiden yola çıkararak, her nokta bu açı

değerini tuttuğu için, bir rota üzerinde iki ardışık noktaların açıları arasındaki farkı bulursak yolun ne kadar dönüşlü olduğunu tahmin etmiş oluruz. Şöyle ki, eğer iki nokta arasındaki açı farkı 40 ve üzeri ise, aracın hızının ciddi bir şekilde (yarı yarıya) azalması gereklidir. Eğer sabit bir mesafede hız azalıyorsa, o yolu katetmeye sarfedilen zaman artar. Dolayısıyla, eğer belirlenen sınırın üzerinde açı farkı var ise, aracın o anda bulunduğu nokta üzerinde önceden tanımlanan zaman etiketi güncellenecektir. Bu durumun algoritması öncesinde olduğu gibi, 'Attributes Table' içinde bulunan 'Field Calculator' kullanılarak hesaplanmıştır. Sonuç olarak, Şekil 6.7de geliştirdiğimiz expression ile dönüş durumu kontrol edilip zaman etiketi güncellenmiştir. Şekil 6.8de, kırmızı karelerle gösterilmiş iki nokta arasında açı farkının 40'ın üzerinde olduğu, ve dolayısıyla da o iki nokta arasındaki mesafeyi daha uzun sürede gittiği gözlemlenebilir.



Şekil 6.6 Noktanın bulunduğu çizgi üzerinde açısal değerinin bulunması

```
--UPDATING TIMESTAMP ACCORDING TO THE ANGLE DIFFERENCE
--Angle difference shows if there is any turn in the route

with_variable('rangle', abs("angle" - attribute(get_feature_by_id(@layer, $id - 1), 'angle')), --
comparing to the previous angle and finding the relative angle
    with_variable('threshold', 40,      -- setting a threshold variable
        with_variable('dist', 40,
            with_variable('start_time', make_datetime(2022, 1, 25, 20, 0, 0), -- setting a start
time variable
                CASE
                    WHEN "distance" = 0
                    THEN @start_time           -- input the start time at zero distance
                ELSE
                    CASE WHEN @rangle < @threshold
                        THEN attribute(get_feature_by_id(@layer, $id-1), 'timestamp') +
to_interval(to_string(@dist / 20) || ' seconds') -- getting the time from the previous row and
adding the interval (distance/speed)

                        ELSE
                            attribute(get_feature_by_id(@layer, $id-1), 'timestamp') + to_interval(
(to_string(@dist / 10) || ' seconds') -- when the angle is above threshold, using a slower speed.
                        END
                    END
                )
            )
        )
    )
)
END
```

Şekil 6.7 Zaman etiketlerinin dönüslere göre güncellenmesi

LAST_POINTS — Features Total: 10100, Filtered: 10100, Selected: 0							
	fid	osm_id	name	type	distance	angle	timestamp
141	141	25302628 Yusuf Aşkin Sokağı		residential	120	40.50520337299...	1/25/2022 20:00:06 (Turkey Standard Time)
142	142	25428552 Kutlugün Sokağı		residential	0	44.99005214101...	1/25/2022 20:00:00 (Turkey Standard Time)
143	143	25428552 Kutlugün Sokağı		residential	40	44.99005214101...	1/25/2022 20:00:02 (Turkey Standard Time)
144	144	25428552 Kutlugün Sokağı		residential	80	43.67973869239...	1/25/2022 20:00:04 (Turkey Standard Time)
145	145	25428552 Kutlugün Sokağı		residential	120	42.40309815715...	1/25/2022 20:00:06 (Turkey Standard Time)
146	146	25428552 Kutlugün Sokağı		residential	160	42.40309815715...	1/25/2022 20:00:08 (Turkey Standard Time)
147	147	25512790 Sultanahmet Meydanı		pedestrian	0	210.7327168770...	1/25/2022 20:00:00 (Turkey Standard Time)
148	148	25512790 Sultanahmet Meydanı		pedestrian	40	93.65109303673...	1/25/2022 20:00:04 (Turkey Standard Time)
149	149	25512790 Sultanahmet Meydanı		pedestrian	80	38.26180266148...	1/25/2022 20:00:08 (Turkey Standard Time)
150	150	25512790 Sultanahmet Meydanı		pedestrian	120	38.28352270393...	1/25/2022 20:00:10 (Turkey Standard Time)
151	151	25512790 Sultanahmet Meydanı		pedestrian	160	38.26790628887...	1/25/2022 20:00:12 (Turkey Standard Time)
152	152	25512790 Sultanahmet Meydanı		pedestrian	200	38.26800677600...	1/25/2022 20:00:14 (Turkey Standard Time)
153	153	25512790 Sultanahmet Meydanı		pedestrian	240	38.27046049670...	1/25/2022 20:00:16 (Turkey Standard Time)
154	154	25512790 Sultanahmet Meydanı		pedestrian	280	38.27571488113...	1/25/2022 20:00:18 (Turkey Standard Time)
155	155	25512790 Sultanahmet Meydanı		pedestrian	320	38.26786881498...	1/25/2022 20:00:20 (Turkey Standard Time)
156	156	25512790 Sultanahmet Meydanı		pedestrian	360	38.27109459554...	1/25/2022 20:00:22 (Turkey Standard Time)
157	157	25512790 Sultanahmet Meydanı		pedestrian	400	38.27109459554...	1/25/2022 20:00:24 (Turkey Standard Time)
158	158	25512790 Sultanahmet Meydanı		pedestrian	440	281.2740163432...	1/25/2022 20:00:28 (Turkey Standard Time)

Şekil 6.8 Noktaların özelliklerini gösteren veriseti

6.2 PostGIS'te 'geom' Sütununa Zaman Boyutunun Eklenmesi

Bir önceki kısımda yapılan işlemlerin sonucunda QGIS'te verinin işlenmesi tamamlanmıştır. Sıradaki adımda veri üzerine yapılacak sorgular için PostGIS fonksiyonlarından faydalandırıldı.

Öncelikle QGIS tarafından 'roads' tablosundan türetilerek elde edilen ve orada işlemlerden geçirilmiş yeni iki veri seti PostGIS'e eklendi. Bu durumda PostGIS tarafından 2 tablo bulunur, bunlar yeni elde ettiğimiz LAST_POINTS ile LAST_ROUTES'tır.

Öncelikle LAST_POINTS tablosunun (Şekil 6.9) yerleşme şekli incelendiğinde şu çıkarımlar yapıldı:

1. Her osm_id değeri bir rotayı temsil eder. Aynı rota üzerindeki noktaların osm_id değerleri aynıdır.
2. Her noktanın kendine özgü bir fid değeri vardır.
3. name, type, distance ve angle bu adımda önemsizdir, daha önce gerekli işlemler için kullanıldılar.
4. timestamp değeri ile geom değeri ayrı sütunlarda bulunmaktadır. Yani geom değerine 3. bir boyut eklenmesine ihtiyaç vardır.

Query Editor Query History Scratch Pad

```
1 select * from public."LAST_POINTS" order by fid
```

Data Output Explain Messages Notifications

	fid	geom	osm_id	name	type	distance	angle	timestamp
			bigint	character varying(4t)	character varying(16)	double precision	double precision	timestamp without ti
1	1	01010000209614000058ABC0620ADD634153D66AD56E535141	4885644	Bab-I Hümayun C...	residential	0	42.78758496576681	2022-01-25 20:00:...
2	2	010100002096140000D1E43BC80DD6341AB83242C76535141	4885644	Bab-I Hümayun C...	residential	40	42.78758496576681	2022-01-25 20:00:...
3	3	010100002096140000866E8D3A11DD6341F017FA697D535141	4885644	Bab-I Hümayun C...	residential	80	45.43740050975379	2022-01-25 20:00:...
4	4	010100002096140000E64388CA14DG341746B4CE6B4535141	4885644	Bab-I Hümayun C...	residential	120	45.43740050975379	2022-01-25 20:00:...
5	5	010100002096140000F9F0C7D8C3E26341ACFD21ED6555141	4885644	Bab-I Hümayun C...	residential	160	42.00650627419994	2022-01-25 20:00:...
6	6	0101000020961400006C4B4DEABFE26341B5FC61AAD5551...	22875933	[null]	motorway_link	0	128.1578340971945	2022-01-25 20:00:...
7	7	010100002096140000F9F0C7D8C3E26341ACFD21ED6555141	22875933	[null]	motorway_link	40	128.1578340971945	2022-01-25 20:00:...
8	8	0101000020961400000F5DCD06C8E263413BF9C78A1555141	22875933	[null]	motorway_link	80	117.1692654027966	2022-01-25 20:00:...
9	9	010100002096140000B633D9A1CCE26341CAE8DFA79D555141	22875933	[null]	motorway_link	120	109.31827847170466	2022-01-25 20:00:...
10	10	01010000209614000013122677D1E2634190F712409B555141	22875933	[null]	motorway_link	160	99.20571961417623	2022-01-25 20:00:...
11	11	0101000020961400005C98505516E36341518D15E982555141	22875933	[null]	motorway_link	0	288.8957699029531	2022-01-25 20:00:...
12	12	01010000209614000062824B9A11E363416C4B222686555141	22875933	[null]	motorway_link	40	288.8957699029531	2022-01-25 20:00:...
13	13	010100002096140000676C46DFCCE3634187092F6389555141	22875933	[null]	motorway_link	80	288.8957699029531	2022-01-25 20:00:...
14	14	0101000020961400004EBABF4008E36341599BEB2F8D555141	22875933	[null]	motorway_link	120	294.37163063748744	2022-01-25 20:00:...

Şekil 6.9 LAST_POINTS tablosundaki değerlerin yerleşme şekli

Sonraki adımda veriye 3 boyutlu indeksleme yapılacağı için bu adımda geom ve timestamp sütunlarını birleştirecek işlemleri yapmayı amaçladık. Bu işlem için PostGIS fonksiyonlarından faydalandık. X, Y ve M koordinatlarını argüman olarak kabul ederek yeni nokta yaratan ST_MakePointM fonksiyonunu kullanmayı seçtik [6].

Bahsedilen fonksiyon argümanlarının hepsini numeric cinsinden kabul ettiği için öncelikle gerekli değerleri doğru formatlara getirdik. X ve Y değerleri ST_X(geom) ve ST_Y(geom) sayesinde çevrilebilir (Şekil 6.10).

	8	SELECT fid, ST_X(geom), ST_Y(geom), osm_id, name 9 from public."LAST_POINTS"				
		Data Output Explain Messages Notifications				
		fid [PK] bigint	st_x double precision	st_y double precision	osm_id bigint	name character varying (48)
1		1	10414163.086019203	4541883.334645825	4885644	Bab-I Hümayun Caddesi
2		2	10414190.257311257	4541912.689728658	4885644	Bab-I Hümayun Caddesi
3		3	10414217.829764616	4541941.655889496	4885644	Bab-I Hümayun Caddesi
4		4	10414246.329133939	4541969.723414291	4885644	Bab-I Hümayun Caddesi
5		5	10414274.022174938	4541998.562074479	4885644	Bab-I Hümayun Caddesi
6		6	10425855.321935378	4544180.418395211	22875933	[null]
7		7	10425886.774406897	4544155.70519964	22875933	[null]

Şekil 6.10 LAST_POINTS tablosunda ST_X(geom) ve ST_Y(geom) değerlerinin gösterimi

Kullanılan diğer SQL sorgusu şekil 6.11de görüldüğü gibidir. Burada EPOCH date time tipinden bir veriyi nümerike çevirmekte kullanılan bir keywordddür. Yani sorgu kısaca date time tipinden bir veriden EPOCH değeri EXTRACT etmeye yarar. Bu EPOCH'ta ulaşmak istediğimiz nümerik değerdir.

	8	SELECT EXTRACT (EPOCH FROM timestamp) from public."LAST_POINTS"				
		Data Output Explain Messages Notifications				
		date_part double precision				
1		1643140800				
2		1643140802				
3		1643140804				
4		1643140806				
5		1643140808				
6		1643140800				
7		1643140802				

Şekil 6.11 SELECT EXTRACT fonksiyonunun test edilmesi

Bu fonksiyon sayesinde timestamp değerleri double precision tipine çevrildi. Daha sonra elde edilen tüm değerler ST_MakePointM fonksiyonunda bir araya getirildi (Şekil 6.12).

```

6   SELECT fid, ST_MakePointM(ST_X(geom), ST_Y(geom),
7   (SELECT EXTRACT (EPOCH FROM timestamp)
8   ), osm_id, name
9   from public."LAST_POINTS"

```

The screenshot shows a PostgreSQL query editor with the following details:

- Query Editor:** Contains the SQL code above.
- Data Output:** Shows the results of the query execution.
- Table Structure:**

fid	st_makepointm	osm_id	name
1	010100004058ABC0620ADD634153D66AD56E53514100000B0157CD841	4885644	Bab-I Hümayun Caddesi
2	0101000040DD1E43BC80DDD6341AB83242C76535141000080B0157CD841	4885644	Bab-I Hümayun Caddesi
3	0101000040866E8D3A11DD6341F017FA697D53514100000B1157CD841	4885644	Bab-I Hümayun Caddesi
4	0101000040E64388CA14DD6341746B4C6E84535141000080B1157CD841	4885644	Bab-I Hümayun Caddesi
5	010100004037A8B54018DD63413C07F9A38B53514100000B21157CD841	4885644	Bab-I Hümayun Caddesi
6	01010000406C4B4DEABFE26341B5FCC61AAD55514100000B0157CD841	22875933	[null]
7	0101000040F9F0C7D8C3E26341ACFD21EDA6555141000080B0157CD841	22875933	[null]
8	0101000040F5DCD06C8E263413BFF9C78A155514100000B1157CD841	22875933	[null]
9	0101000040B633D9A1CCE26341CAE8DFA79D555141000080B1157CD841	22875933	[null]

Şekil 6.12 ST_MakePointM fonksiyonunun test edilmesi

Yeni elde edilen geom değerleri LAST_POINTS tablosunun gerçek geom değerleriyle karşılaştırıldığında işlemde başarılı olunduğu görüldü. Sonraki adımda bu sorgunun sonucu olarak gelen tablo indexleme ve analizler için yeni bir tabloya insert edildi. Bu işlemin sebebi daha önce de belirtildiği gibi indexlemeyi 3 boyutlu geom değeri üzerinden yapmaktadır. Burada da eski tabloyu bozmamak için 3 boyutlu geomları tutacak yeni bir tablo oluşturuldu. Sorgunun sonucu yeni tabloya eklenip orada indexlenmesi amaçlandı. Bu sebeple tablonun bütün sütunları aynen yeni tabloya kopyalandı. Ancak eski geom sütunu POINT türünden 2 boyutlu olduğu için, ve ihtiyaç duyulan sütun POINTM tipinden 3 boyutlu olduğu için resimde görülen sorgu ile önce var olan geom sütunu silindi, daha sonra ihtiyaca uygun yeni bir sütun tanımlandı. Bu işlem için PostGIS'in AddGeometryColumn fonksiyonu kullanıldı (Şekil 6.13).

```

1
2 create table points_3dgeom as (select * from public."LAST_POINTS");
3
4 alter table public.points_3dgeom drop column geom
5
6 SELECT AddGeometryColumn
7 ('public','points_3dgeom','geom',4326,'POINTM',3);
8
9

```

The screenshot shows a PostgreSQL query editor with the following details:

- Query Editor:** Contains the SQL code above.
- Data Output:** Shows the results of the query execution.
- Table Structure:**

addgeometrycolumn
text

Şekil 6.13 AddGeometryColumn fonksiyonunun çalıştırılması

Artık tüm kayıtlar yeni geomlar ile birlikte points_3dgeom tablosuna aktarılabilir durumdadır (Şekil 6.14).

```
10  insert into points_3dgeom (
11    select fid, osm_id, name, type, distance, angle, timestamp,
12      ST_MakePointM(ST_X(geom), ST_Y(geom),
13      (SELECT EXTRACT (EPOCH FROM timestamp))
14    )
15  from public."LAST_POINTS")
```

Data Output Explain Messages Notifications

```
INSERT 0 10100
```

```
Query returned successfully in 58 msec.
```

Şekil 6.14 insert into ile yeni tabloya kayıtların eklenmesi

İki tablo karşılaştırıldığında geom değerleri arasındaki farklar görülebilir (Şekil 6.15).

17 select fid, name, ST_AsText(geom) from points_3dgeom				
Data Output Explain Messages Notifications				
	fid bigint	name character varying (48)	st_astext text	
1	1	Bab-I Hümayun Caddesi	POINT M (10414163.086019203 4541883.334645825 1643140800)	
2	2	Bab-I Hümayun Caddesi	POINT M (10414190.257311257 4541912.689728658 1643140802)	
3	3	Bab-I Hümayun Caddesi	POINT M (10414217.829764616 4541941.655889496 1643140804)	
4	4	Bab-I Hümayun Caddesi	POINT M (10414246.329133939 4541969.723414291 1643140806)	
5	5	Bab-I Hümayun Caddesi	POINT M (10414274.022174938 4541998.562074479 1643140808)	
6	6	[null]	POINT M (10425855.321935378 4544180.418395211 1643140800)	
7	7	[null]	POINT M (10425886.774406897 4544155.70519964 1643140802)	
8	8	[null]	POINT M (10425920.212568788 4544133.884582336 1643140804)	
9	9	[null]	POINT M (10425957.057763916 4544118.623041341 1643140806)	

Şekil 6.15 Yeni tablonun son hali

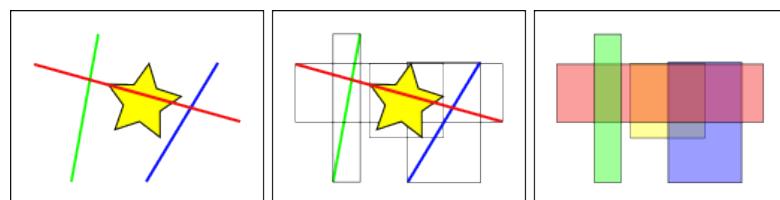
Projede veriye uygulamamız gereken tüm işlemler tamamlandı. Bu adımdan sonra indexlemeler ve performans analizleri yapıldı.

7

Deneysel Sonuçlar

7.1 İndexleme İşleminin Açıklaması ve Kullanılma Sebepleri

Verilerin indexlenmesi işlemi, özellikle büyük yer kaplayan veri tabanlarında sorguların yapılabilmesi için büyük fayda sağlar. İndexleme yapılmaması, her sorguda verilerin lineer aramadan geçirilmesi gerekiirdi. Normal veri tabanlarında indeksleme işlemleri, bu iş için seçilen sütunun değerleri üzerinden bir ağaç veri yapısına yerleştirilerek yapılır. Ancak uzamsal veri tabanlarında böyle bir uygulama yapmak mümkün değildir. Geometrik özellikler indekslenebilir değerler olmadığı için onların yerine genellikle şekilde belirtildiği gibi sınırlayıcı kutular oluşturulur ve daha sonra elde edilen bu sınırlayıcı kutular uzamsal veri yapılarına uygun veri yapılarına yerleştirilir (Şekil 7.1) [7].



Şekil 7.1 Bounding Boxların yerleşimi

Değişik ihtiyaçlara uyum sağlayabilmek için farklı veri yapıları ile verilerini yerleştiren index tipleri üretilmiştir. PostGIS'in geometri türüne uyum sağlayan ve günümüzde hala uygulamalar tarafından desteklenmekte olan 10 farklı tür index vardır.

Geometri sütunlarına eklenebilecek olan indexler temelde 3 grupta toplanabilir: bunlar GIST, BRIN ve SPGIST'tir. İndexledikleri verilerin boyutlarına göre 2 boyutlu GIST indexleri, 3 boyutlu GIST indexleri gibi alt gruplara da ayrırlırlar. Projemizde kullandığımız veri 3 boyutlu ve POINTM tipinde olduğu için sadece 3 boyutlu tipler kullanıldı [8].

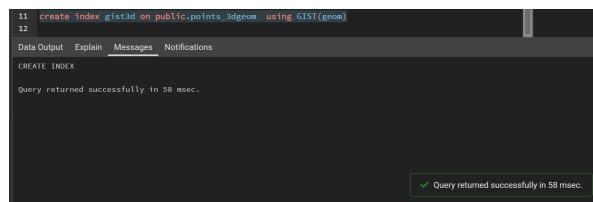
7.2 Kullanılan Indexler

Bu kısımda, kullanılacak olan indexler hakkında zaten bilinen belli özelliklerden kısaca bahsedilecektir, daha sonra performans analizlerinde bu beklentiler incelenecektir. Indexlerin yüklenme sürelerinden de bahsedilecektir.

7.2.1 GIST

GIST metodu r-tree veri yapısını implement eden index tipidir. G harfi ‘generalized’ anlamına gelir, GIST metodu da en genel ihtiyaçlara cevap vermek için tasarlanmıştır, en çok uniform dağılmış veri tiplerine uygundur, bu nedenle özelleştirilmiş tablolarda yeterince verimli olmadığı bilinmektedir.

Indexin build süresi şekil 7.2deki gibidir.



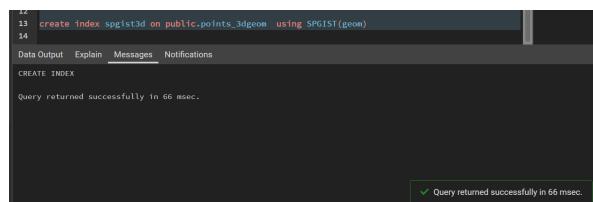
```
11 create index gist3d on public.points_3dgeom using GIST(geom)
12
13
14 Data Output Explain Messages Notifications
15 CREATE INDEX
16
17 Query returned successfully in 58 msec.
```

Şekil 7.2 Create Index Using GIST sorgusu

7.2.2 SPGIST

SPGIST metodu, GIST metodundan yola çıkılarak dizayn edilmiştir ve onun kadar genelleyici özelliklere sahip bir index tipidir SP kısmı Space Partitioning anlamına gelir. Ancak GIST metodundan farklı olarak üst üste gelen bounding boxlar arası karışıklıkları çözmeye odaklanmıştır. Quad-tree veri yapısı kullanır.

Indexin build süresi şekil 7.3deki gibidir.



```
14
15 create index sppist3d on public.points_3dgeom using SPGIST(geom)
16
17 Data Output Explain Messages Notifications
18 CREATE INDEX
19
20 Query returned successfully in 66 msec.
```

Şekil 7.3 Create Index Using SPGIST sorgusu

7.2.3 BRIN

BRIN metodu ise daha önce bahsettiğimiz diğer iki metottan tamamen farklıdır. Block-Range teknigi ile tabloları indexler, bu işlem tabloları belli bloklara ayırip o

bloklar içerisindeki en büyük ve en küçük elemanı kaydeden ve bunlar üzerinden ağazının dallarını gezen bir yapıdır. Bu nedenle sıralı verilerde efektif olan bir yöntem olarsk bilinmektedir.

Indexin build süresi şekil 7.4deki gibidir.

```
'  
8  
9  create index brin3d on public.points_3dgeom  using BRIN(geom)  
10  
Data Output Explain Messages Notifications  
CREATE INDEX  
Query returned successfully in 52 msec.
```

Şekil 7.4 Create Index Using BRIN sorgusu

8

Performans Analizleri ve Yapılan Çıkarımlar

Bu kısımda performans analizleri yapılırken göz önünde bulundurulan şartlardan ve incelenen tablo yapısından bahsedilecektir. Daha sonra analizlerin sonuçları açıklanacaktır ve grafikler yardımı ile görselleştirilecektir.

8.1 Analizler Yapılırken Dikkat Edilen Faktörler

Bu sonuçların belli şartlar altında elde edildiğinin belirtilmesi gereklidir. Öncelikle tablo büyülüğu kullanılan bilgisayarların CPU gücünün kısıtları içerisinde küçük tutulmaya çalışılmıştır. Bunun dışında PostgreSQL belli durumlarda daha hızlı işlem yapabileceğini hesaplayıp, kullanıcının kontrolü dışında bir şekilde indexlemeyi kullanmamayı tercih etmektedir. Bu seçimi bazen sorgu sonucunun cache’te bulunması ya da kullanıcının erişemediği daha kullanışlı indexlerin bulunması gibi değişik faktörler etkileyebilir. Çalışmamız bizi ilgilendiren index tiplerinin özellikleri ve hangi durumlarda faydalı olabileceğini incelemek üzerine olduğundan, sonuçlar alınırken olabildiğince bu faktörlerden ayrı tutulmaya çalışılmıştır.

8.2 Analizi Yapılan Tablo

Bu kısımda points_3dgeom tablosu üzerinde çalışılacaktır. Tablonun özellikleri incelenirse analizlerin sonuçları hakkında tahminlerde bulunulabilir. Şekil 8.1de tablonun geom değerlerine göre sıralandığı sorgu sonucu görülmektedir.

Tablo üzerinde dikkat edilmesi gereken bazı noktalar şunlardır:

- Tablo geomertisi POINTM tipinde olan noktalardan oluşur.
- X ve Y değerleri dışında M boyutunu da taşır. Bu boyut zamanı temsil eder.
- Noktalar uniform dağılmaktadır.

Data Output										Explain	Messages	Notifications
	fid bigint	osm_id bigint	name character varying	type character varying	distance double precision	angle double precision	timestamp timestamp without time zone	geom geometry				
1	8462	719901925	[null]	track	0	342.6022144875666	2022-01-25 20:00:00	0101000060E61000000DFFC167471E46341CFCA419C9F60514100000B0157CD841				
2	496	35037404	76_Sokak	residential	40	178.57943939520715	2022-01-25 20:00:02	0101000060E61000001570A05BABA363417AD41F7AAD5F514100000B0157CD841				
3	497	35037404	76_Sokak	residential	80	198.55120641195347	2022-01-25 20:00:04	0101000060E61000009285490AAE363413B864ABBA35F514100000B1157CD841				
4	495	35037404	76_Sokak	residential	0	160.10692469661555	2022-01-25 20:00:00	0101000060E61000001DB4EB2DAAE36341EA1FBC13B75F514100000B0157CD841				
5	6803	230569919	[null]	residential	120	16.648737018277313	2022-01-25 20:00:06	0101000060E6100000F87052B86E36341C388F6881B60514100000B1157CD841				
6	6802	230569919	[null]	residential	80	11.549566218374615	2022-01-25 20:00:04	0101000060E6100000B6F8026A84E36341EB10CB381260514100000B1157CD841				
7	6800	230569919	[null]	residential	0	34.202200559050404	2022-01-25 20:00:00	0101000060E6100000087A1A80E363418E6A9D310060514100000B0157CD841				
8	6801	230569919	[null]	residential	40	12.6126163058583	2022-01-25 20:00:02	0101000060E61000000357E44CB3E36341FD663D790860514100000B0157CD841				
9	6804	230569919	[null]	residential	160	11.819352809280776	2022-01-25 20:00:00	0101000060E6100000078E8450B7E363412C1704A32360514100000B2157CD841				
10	6805	230569919	[null]	residential	200	39.15772767722611	2022-01-25 20:00:10	0101000060E61000000A7830E689E36341BE783CB12D0514100000B02157CD841				
11	8464	719901925	[null]	track	80	24.91221698308642	2022-01-25 20:00:06	0101000060E6100000A21F3A72E463415F3EB82B260514100000B1157CD841				
12	8465	719901925	[null]	track	120	61.12671269025888	2022-01-25 20:00:08	0101000060E610000004891FBDB75E46341FAE26BD860514100000B02157CD841				
13	8463	719901925	[null]	track	40	353.10678524375276	2022-01-25 20:00:02	0101000060E61000000221F6F04D804E36341C2F96659BF5F514100000B0157CD841				
14	6869	230921254	[null]	residential	0	267.35085958960025	2022-01-25 20:00:00	0101000060E6100000A96004D804E36341C2F96659BF5F514100000B0157CD841				

Şekil 8.1 points_3dgeom tablosunun geom değerlerine göre sıralı sorgusunun sonucu

- osm_id değeri aynı rotada bulunan noktalar için ortak bir değerdir, fid değeri ise noktaların oluşturulma sırasına göre 1 den 10100. satırda kadar sıralı giden bir değerdir. Primary key görevi taşır.
- ORDER BY geom sorgusunun ekran görüntüsünde görüldüğü üzere noktalar sıralı değildir. Bu fid değerlerinin karışık yerleşmesinden anlaşılabilir.

8.3 Explain Analyze Sorgusu İle Çıkarımlar

Bu kısımda verinin indexsiz sorgulanmasıyla, üç farklı tipinde indexlendikten sonra sorgulanmaları karşılaştırılacaktır. Karşılaştırmada EXPLAIN ANALYZE sorgularının sonuçları göz önünde bulundurulmuştur. Sorgu sonuçlarından elde edilen üç farklı değer grafiklerle görselleştirilmiştir. Bunlar planning time, execution time ve total runtimedir.

Ayrıca uzamsal indexlerin çalışması gereken diğer bir şartta uzamsal fonksiyonlar ile oluşturulmuş sorguların kullanılmasıdır.

Bu durumda POINTM tipi üzerinde çalışan çok fazla fonksiyon olmadığı için argüman olarak verilen bir noktaya yine argüman olarak verilen bir uzaklıkta olan diğer noktaları döndüren ST_3DDWithin PostGIS fonksiyonu kullanılmıştır.

8.3.1 Indexsiz

Şekil 8.2de index yapılmadan önce sorgunun EXPLAIN ANALYZE sonucunu görüyoruz. Dikkat çeken kısım sequential scan yapıldığıdır. Ayrıca sorgunun planning

time'ının neredeyse olmadığı gözlemlenebilir. Sequential search yapılan bir sorguda planlamaya gerek yoktur.

```

Query Editor Query History
Scratch P. ✎
1 explain analyze
2 select name, ST_AsText(geom)
3 from points_3dgeom
4 where ST_3DDWithin(
5     points_3dgeom.geom,
6     ST_GeomFromEWKT('SRID=4326;POINT M (10414163.086019203 4541883.334645825 1643140800)'), 
7     2589)
8

Data Output Explain Messages Notifications
QUERY PLAN
text
1 Seq Scan on points_3dgeom (cost=0.00..253109.25 rows=1 width=50) (actual time=0.392..396.867 rows=135 loops=1)
2 [...] Filter: st_3ddwithin(geom, '0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841'::geometry, '2589'::double precision)
3 [...] Rows Removed by Filter: 9965
4 Planning Time: 0.199 ms
5 Execution Time: 396.925 ms

Successfully run. Total query runtime: 644 msec. 5 rows affected.

```

Şekil 8.2 ST_3DDWithin ile indexsiz EXPLAIN ANALYZE sorgusu

8.3.2 GIST

Şekil 8.3de GIST indexi uygulandıktan sonra yapılan bir sorgunun EXPLAIN ANALYZE sonucunu görüyoruz. Bu sefer sequential scan yapılmamıştır. GIST bir ağaç yapısında olduğu için PostgreSQL'in gerekli ağaç dallarına ulaşırken gittiği yol gözlemlenebilir. Indexsiz sorguya kıyasla gözle görülür bir planning time farkı vardır. Fakat execution time çok büyük bir farkla azalmıştır.

```

Query Editor Query History
Scratch P. ✎
1 explain analyze
2 select name, ST_AsText(geom)
3 from points_3dgeom
4 where ST_3DDWithin(
5     points_3dgeom.geom,
6     ST_GeomFromEWKT('SRID=4326;POINT M (10414163.086019203 4541883.334645825 1643140800)'), 
7     2589)
8

Data Output Explain Messages Notifications
QUERY PLAN
text
1 Bitmap Heap Scan on points_3dgeom (cost=9.90..5810.23 rows=1 width=50) (actual time=0.187..6.202 rows=135 loops=1)
2 [...] Filter: st_3ddwithin(geom, '0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841'::geometry, '2589'::double precision)
3 [...] Rows Removed by Filter: 86
4 [...] Heap Blocks: exact=24
5 [...] -> Bitmap Index Scan on threedindex (cost=0.00..9.89 rows=216 width=0) (actual time=0.082..0.082 rows=221 loops=1)
6 [...] Index Cond: (geom && st_expand('0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841'::geometry, '2589'::double precision))
7 Planning Time: 3.510 ms
8 Execution Time: 6.285 ms

Successfully run. Total query runtime: 76 msec. 8 rows affected.

```

Şekil 8.3 ST_3DDWithin ile GIST indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu

8.3.3 SPGIST

Şekil 8.4de SPGIST indexi uygulandıktan sonra yapılan bir sorgunun EXPLAIN ANALYZE sonucunu görüyoruz. Bu sorguda da ağaç yapısında erişilen dallar ayrıntılı bir şekilde gösterilmiştir. Üstelik GIST'te verilen bilgilerle kıyaslandığında aynı yolu izledikleri gözlemlenebilir. Bu iki yöntemin çok benzer yapıda olmasından kaynaklıdır. GIST ve SPGIST arasında süre farkları çok az olduğu için etkilerini gözlemlemek güçtür. Ancak SPGIST'in neredeyse aynı zamanı planning'de harcadığı, execution'ı GIST'e göre daha bile kısa tuttuğu söyleylenebilir.

The screenshot shows the PostgreSQL Query Editor interface. In the top-left, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 explain analyze
2 select name, ST_AsText(geom)
3 from points_3dgeom
4 where ST_3DDWithin(
5     points_3dgeom.geom,
6     ST_GeomFromEWKT('SRID=4326;POINT M (10414163.086019203 4541883.334645825 1643140800)'),
7     2589)
```

In the bottom-right corner of the editor window, there is a green success message: "Successfully run. Total query runtime: 73 msec. 8 rows affected."

Below the Query Editor, the 'Data Output' tab is selected, showing the 'QUERY PLAN' section:

text
1 Bitmap Heap Scan on points_3dgeom (cost=9.90..5810.23 rows=1 width=50) (actual time=0.138..4.284 rows=135 loops=1)
2 [...] Filter: st_3ddwithin(geom, 0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841::geometry, '2589':double precision)
3 [...] Rows Removed by Filter: 86
4 [...] Heap Blocks: exact=24
5 [...] > Bitmap Index Scan on threediindex (cost=0.00..9.89 rows=216 width=0) (actual time=0.042..0.043 rows=221 loops=1)
6 [...] Index Cond: (geom && st_Expand('0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841'::geometry, '2589':double precision))
7 Planning Time: 3.844 ms
8 Execution Time: 4.327 ms

Şekil 8.4 ST_3DDWithin ile SPGIST indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu

8.3.4 BRIN

Şekil 8.5de BRIN indexi uygulandıktan sonra yapılan bir sorgunun EXPLAIN ANALYZE sonucunu görüyoruz. Bu sorguda BRIN indexinin diğer ikisinden ne kadar farklı olduğu açıkça görülebilir. Ağaç yapısı hakkında bilgi verilmemiştir. PostgreSQL sequential scan yapmayı tercih etmiştir. Şekil 8.1e geri dönülürse BRIN indexinin işleri bu kadar kötülestirmesinin sebebi anlaşılabılır. Bu index sıralı ve düzenli verilerde faydalıdır, çünkü veriyi bloklara ayırıp her bloktaki en büyük ve küçük değerleri kaydeder, sorgu sonuçlarına bu değerler üzerinden erişerek çalışır. ORDER BY geom sorgusuyla tablonun satırlarının karışması elimizdeki verinin bu indexe uygun olmadığını göstergesidir.

8.4 Grafikler ve Çıkarımlar

Bu kısımda şimdije kadar elde edilen analizlerin sonuçları matplotlib kütüphanesi ile görselleştirilmiştir.

```

Query Editor  Query History
1 explain analyze
2 select name, ST_AsText(geom)
3 from points_3dgeom
4 where ST_3DDWithin(
5     points_3dgeom.geom,
6     ST_GeomFromEWKT('SRID=4326;POINT M (10414163.086019203 4541883.334645825 1643140800)'),
7     2589)
8

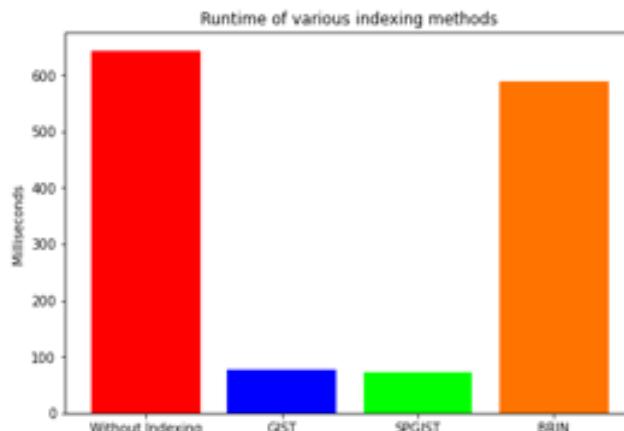
Data Output Explain Messages Notifications
QUERY PLAN
text
1 Seq Scan on points_3dgeom (cost=0.00..253109.25 rows=1 width=50) (actual time=1.171..401.990 rows=135 loops=1)
2 [...] Filter: st_3ddwithin(geom, '0101000060E610000058ABC0620ADD634153D66AD56E535141000000B0157CD841'::geometry, '2589'::double precision)
3 [...] Rows Removed by Filter: 9965
4 Planning Time: 1.894 ms
5 Execution Time: 402.047 ms

Successfully run. Total query runtime: 589 msec. 5 rows affected.

```

Şekil 8.5 ST_3DDWithin ile BRIN indexi yapıldıktan sonra EXPLAIN ANALYZE sorgusu

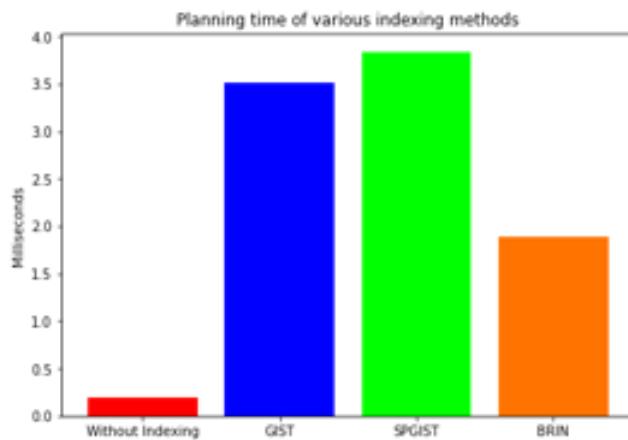
Şekil 8.6de görüldüğü gibi total runtimelar karşılaştırıldığında indexsiz ve BRIN arasında ufak bir fark vardır. GIST ve SPGIST diğer ikisine kıyasla oldukça verimli gözükmemektedir. Aralarında runtime açısından çok bir fark yoktur.



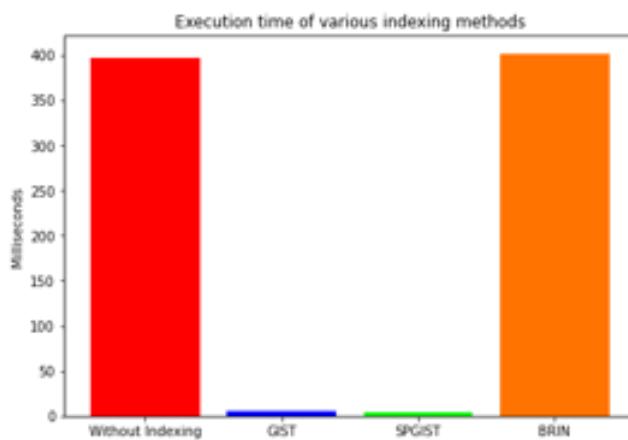
Şekil 8.6 EXPLAIN ANALYZE sorgularında total runtimelerin karşılaştırılması

Şekil 8.7de her index için farklı planning timelar gözlemlenebilir. Burada en çok planning time'in SPGIST tarafından harcadığı dikkat çekicidir. GIST yine yakın bir değer göstermiştir ancak planlamaya SPGIST'ten daha az vakit harcadığı açıktır. BRIN indexi de verimli olmamasına rağmen bu adıma oldukça fazla zaman harcamıştır. Ve indexsiz sequential scan sorgusu doğal olarak plan yapmadan ilerlemiştir.

Son olarak şekil 8.8de execuiton time lar karşılaştırılabilir. Burada can alıcı nokta indexsiz ve BRIN indexli sorguların çok yüksek olduğu ve SPGIST ile GIST'in neredeyse hiç zaman harcanmamış gibi gözükmektedir. Ufak bir farkla BRIN indexinin işleri normal sorgudan bile daha kötü aradığı görülebilir.



Şekil 8.7 EXPLAIN ANALYZE sorgularında planning timeların karşılaştırılması



Şekil 8.8 EXPLAIN ANALYZE sorgularında execution timeların karşılaştırılması

Bu durumda şu çıkarımlar yapılır:

- SPGIST en faydalı yöntemdir, ancak GIST'te oldukça uygundur. Verimizin uniform dağıldığı önceden bilinmekte olduğundan bu beklenen bir sonuçtır.
- Tablomuz sırasız olduğu için BRIN indexi işe yaramamıştır.
- SPGIST, GIST'e göre daha geliştirilmiş bir yöntemdir. Bu nedenle planning'de daha çok zaman harcamıştır, fakat sonuçta sorgu daha çabuk execute edilmiş ve runtime daha azdır.

9 Sonuç

Projemizde uzamsal veritabanlarına giriş yaptık ve yeni konseptler öğrenmiş olduk. Bu alanda yaygın olarak kullanılan toolları tanıdık ve veritabanı kullanabilme becerilerimizi geliştirdik.

Projemizin ilk yarısında bulunan açık kaynak üzerinden İstanbul haritasının çeşitli katmanları çekildi ve QGIS ortamına eklendi. Bu sayede, sentetik veriseti üretimi için gerekli ortam hazırlanmış oldu. Yol haritasının CRS formatı ve ölçü birimi değiştirildi, rasgele 2500 tane sokak seçildi ve bunlardan belirli bir uzunluğun altında olanlar filtreldi. Devamında, Bu rotalar üzerinde her 40 metreden bir, rota üzerinde hareket eden araçların yörungesini temsil eden noktalar üretildi ve sokakların keskin dönüşlü yerleri de göz önünde bulunarak belirli bir hız'a göre zaman etiketleri atandı.

İkinci yarıda QGIS'te işlenen verileri PostGIS'e taşındı ve önceden 2 boyutlu olan geometri sütununa zaman boyutunu ekleme işlemi tamamlandı. Elde edilen son tablo üzerinde öğrendiğimiz 'spatial indexing' yöntemleri sayesinde analizler yapıldı. Analizler sonucunda PostGIS fonksiyonlarıyla oluşturulmuş sorgular kullanıldı. Sonuçta, büyük ve 3 boyutlu verilerde indekslemenin faydalı olduğu yeniden kanıtlanmış oldu. Alınan sonuçlar da Python'un Matplotlib kütüphanesi yardımıyla görselleştirildi. EXPLAIN ANALYZE sorguları yorumlandı.

Referanslar

- [1] W. J. Cromartie, “Oos 23-5: Long-term ecological research at an undergraduate college: Student participation in monitoring and assessing ecosystem change,” in *The 93rd ESA Annual Meeting*, 2008.
- [2] R. S. Liévanos, “Racialised uneven development and multiple exposure: Sea-level rise and high-risk neighbourhoods in stockton, ca,” *Cambridge Journal of Regions, Economy and Society*, vol. 13, no. 2, pp. 381–404, 2020.
- [3] J. Hu, H. Tang, K. C. Tan, and H. Li, “How the brain formulates memory: A spatio-temporal model research frontier,” *IEEE Computational Intelligence Magazine*, vol. 11, no. 2, pp. 56–68, 2016.
- [4] QGIS Documentation: *Working with Projections*, https://docs.qgis.org/3.16/en/docs/user_manual/working_with_projections/working_with_projections.html, [Online; accessed 21-Jan-2022], 2020.
- [5] QGIS Documentation: *Random Extract*, https://docs.qgis.org/3.16/en/docs/user_manual/processing_algs/qgis/vectorselection.html#random-extract, [Online; accessed 21-Jan-2022], 2020.
- [6] PSC, PostGIS Documentation: *8.4. Geometry Constructors: ST_MakePointM*, https://postgis.net/docs/manual-2.2/ST_MakePointM.html, [Online; accessed 22-Jan-2022].
- [7] M. L. Paul Ramsey, PostGIS Workshop: *Spatial Indexing*, <https://postgis.net/workshops/postgis-intro/indexing.html#spatial-indexing>, [Online; accessed 22-Jan-2022], 2012.
- [8] P. Ramsey, *(The Many) Spatial Indexes of PostGIS*, <https://blog.crunchydata.com/blog/the-many-spatial-indexes-of-postgis>, [Online; accessed 20-Jan-2022], 2021.

Özgeçmiş

BİRİNCİ ÜYE

İsim-Soyisim: Ramiz Mammadlı

Doğum Tarihi ve Yeri: 11.06.1999 Nahçıvan, Azerbaycan

E-mail: l1118903@std.yildiz.edu.tr

Telefon: 0553 896 83 34

Staj Tecrübeleri:

İKİNCİ ÜYE

İsim-Soyisim: Elif Yağmur Duran

Doğum Tarihi ve Yeri: 26.07.1999 İstanbul, Türkiye

E-mail: l1118071@std.yildiz.edu.tr

Telefon: 0537 820 70 66

Staj Tecrübeleri: "Experian Limited" Şirketi, Analiz Departmanı

Proje Sistem Bilgileri

Sistem ve Yazılım: Windows İşletim Sistemi, Python, PostgreSQL, POSTGIS, QGIS

Gerekli RAM: 8 GB

Gerekli Disk: 256 MB