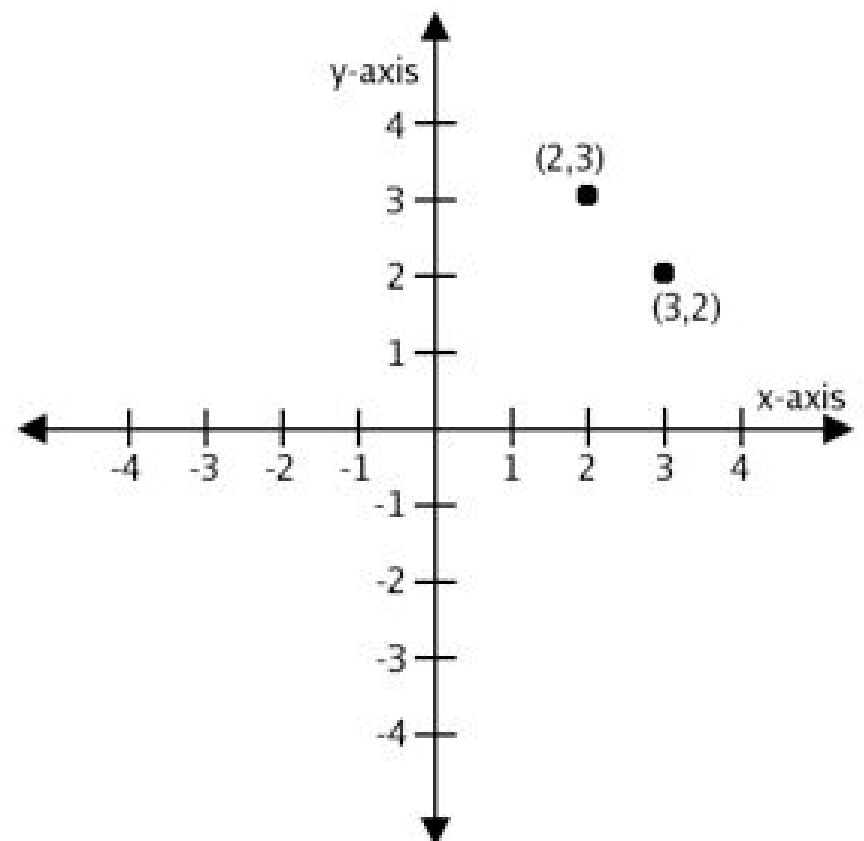


# **GEOGRAFIA POSTGIS**

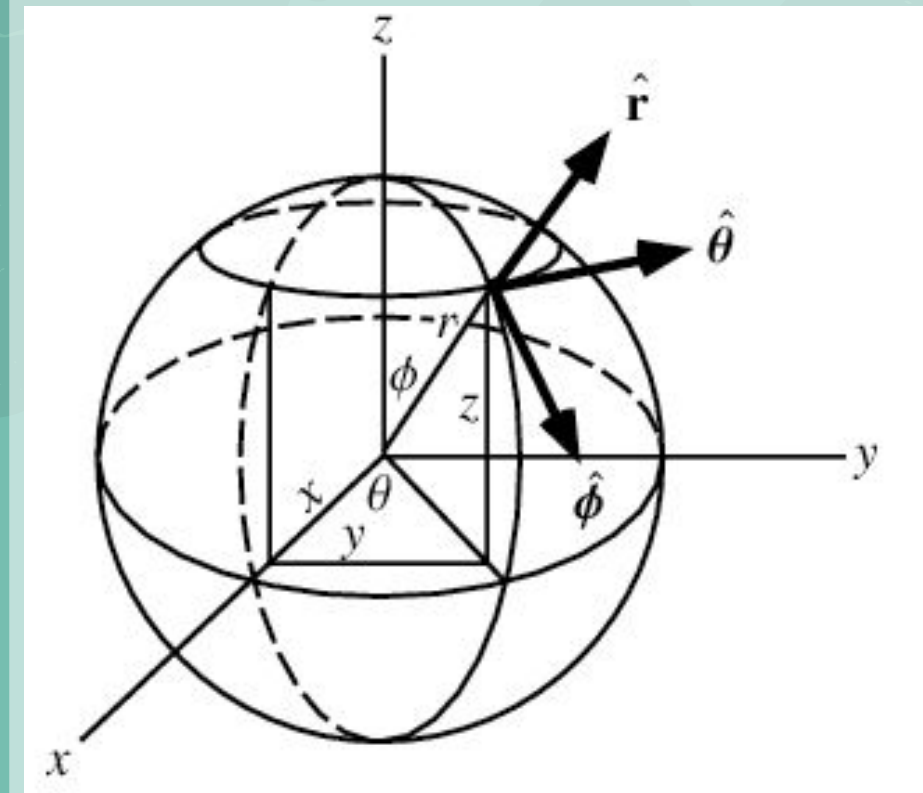


# COORDENADAS

## CARTESIANAS



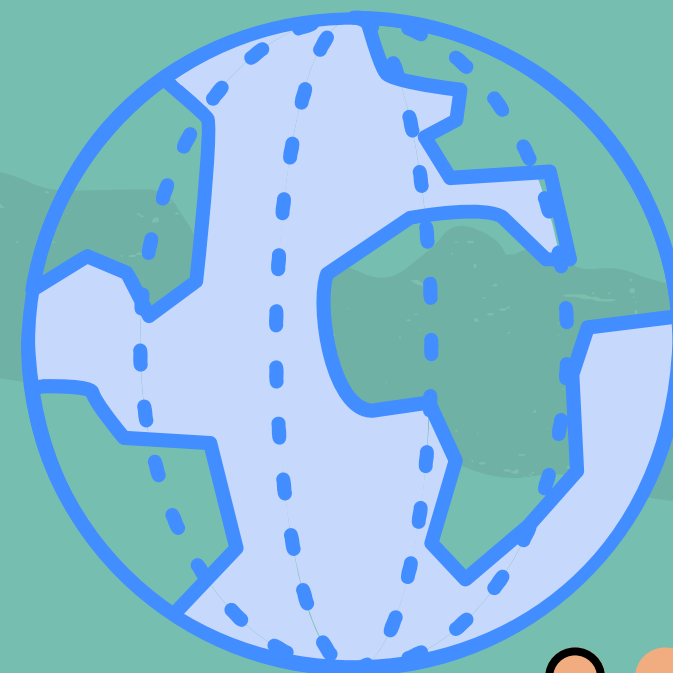
## ESFERICAS



# ¿Qué son las coordenadas geográficas?

Las coordenadas geográficas (latitud y longitud) no son cartesianas, sino angulares. Es decir:

- Latitud: ángulo desde el ecuador (horizontal).
- Longitud: ángulo desde el meridiano de Greenwich (vertical).





# ¿Por qué no usar coordenadas geográficas directamente como cartesianas?

Si tratas latitud/longitud como si fueran coordenadas planas:

- 01** Las distancias serán erróneas (los grados no son unidades constantes).
- 02** Las áreas, intersecciones o inclusiones serán imprecisas.
- 03** ST\_Distance(geometry) devuelve distancia en grados, no metros.

```
1 SELECT ST_Distance(  
2     ST_SetSRID(ST_PointFromText(ST_AsText(a.geom)), 4326),  
3     ST_SetSRID(ST_PointFromText(ST_AsText(b.geom)), 4326)  
4 ) AS distancia_grados  
5 FROM  
6     "Incidentes_2024_Tunjuelito_4686_2" a,  
7     "Incidentes_2024_Tunjuelito_4686_2" b  
8 WHERE  
9     a.id = 1 AND  
10    b.id = 2;
```

	distancia_grados double precision	
1	0.041285608902426246	

# ¿Qué hace diferente al tipo geography?


**PostGIS ofrece dos tipos espaciales:**

- geometry: plano cartesiano (proyección).
- geography: esférico (globo terrestre).

**Con geography, los cálculos:**

- Usan una esfera o esferoide real.
- Devuelven resultados en metros.
- Son útiles para datos que abarcan grandes distancias.

```
SELECT ST_SRID(geom)
FROM "Incidentes_2024_Tunjuelito_4686_2"
LIMIT 1;
```

	st_srid 
1	4686

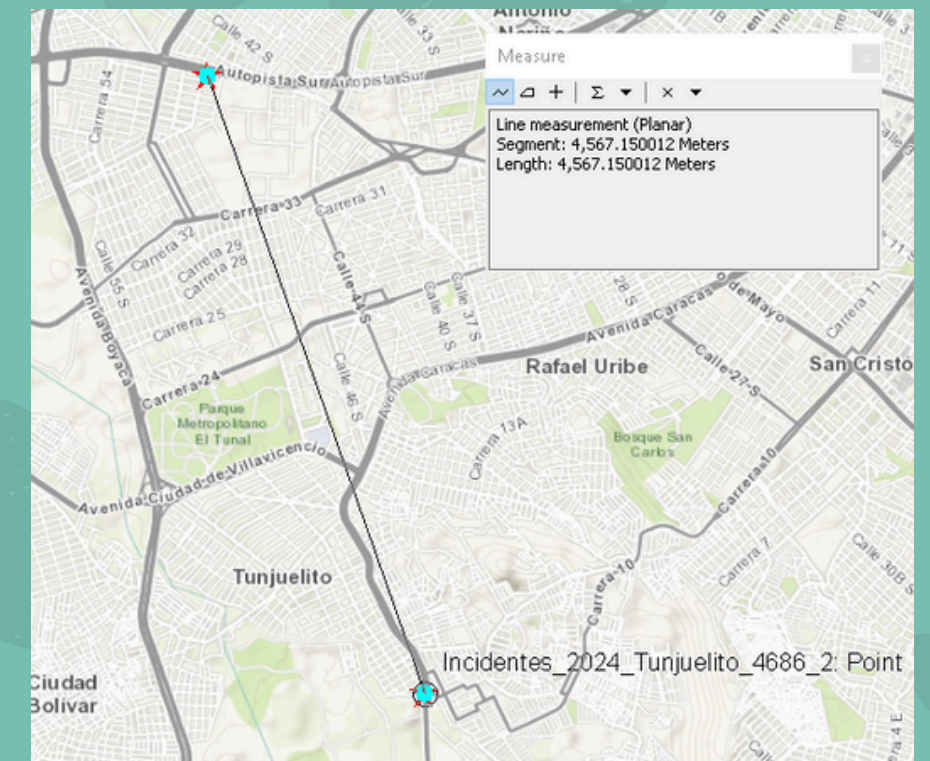


# ¿Y si quiero medir cercanía a una ruta o punto?

Rutas ortodrómicas (en una esfera) son más precisas:

```
SELECT ST_Distance(  
  a.geom::geography,  
  b.geom::geography  
) AS distancia_metros  
FROM  
  "Incidentes_2024_Tunjuelito_4686_2" a,  
  "Incidentes_2024_Tunjuelito_4686_2" b  
WHERE  
  a.id = 1 AND  
  b.id = 2;
```

	distancia_metros double precision 
1	4567.14820934





# Uso práctico de geography

Crear una tabla:

```
CREATE TABLE incidentes_geog AS  
SELECT  
geom::geography AS geog,  
fecha,  
estacion  
FROM "Incidentes_2024_Tunjuelito_4686_2";
```

Construir un índice espacial en una tabla de geografía es exactamente lo mismo que para la geometría:

```
CREATE INDEX incidentes_geog_gix  
ON incidentes_geog  
USING GIST (geog);
```

# Uso práctico de geography

Aquí tienes una consulta para encontrar todas las estaciones de metro a 500 y su dirección en azimut

```
WITH punto_referencia AS (  
    SELECT 'SRID=4326;POINT(-74.123 4.567) '::geography AS geog  
)  
SELECT  
    i.estacion,  
    i.fecha,  
    ST_Distance(  
        pr.geog,  
        ST_Transform(i.geog::geometry, 4326)::geography  
    ) AS distancia_metros,  
    degrees(  
        ST_Azimuth(  
            pr.geog,  
            ST_Transform(i.geog::geometry, 4326)::geography  
        )  
    ) AS direccion_grados  
FROM incidentes_geog i,  
    punto_referencia pr  
WHERE ST_DWithin(  
    pr.geog,  
    ST_Transform(i.geog::geometry, 4326)::geography,  
    500  
):
```

	estacion character varying (254)	fecha date	distancia_metros double precision	direccion_grados double precision
1	B-11 CANDELARIA	2024-02-06	497.78603702	215.19208319515712
2	B-11 CANDELARIA	2024-03-28	398.54968181	233.76275921634286
3	B-11 CANDELARIA	2024-04-13	409.05075259	240.13525391839158
4	B-11 CANDELARIA	2024-05-25	494.89835637	223.391426829041
5	B-10 MARICHUELA	2024-10-30	398.54871889	233.76219451358506
6	B-11 CANDELARIA	2024-11-05	418.96864549	218.4547186393802





# geography y geometry

Cuando necesitas funciones que solo existen para geometry:

```
SELECT
  ST_Distance(
    i1.geom,          -- tipo geometry (distancia plana en grados/proyección)
    i2.geom
  ) AS geometry_distance,

  ST_Distance(
    i1.geom::geography, -- tipo geography (distancia en metros, sobre elipsoide)
    i2.geom::geography
  ) AS geography_distance

FROM
  "Incidentes_2024_Tunjuelito_4686_2" i1,
  "Incidentes_2024_Tunjuelito_4686_2" i2
WHERE
  i1.id = 1 AND
  i2.id = 2;
```






	geometry_dstance double precision	geography_dstance double precision
1	0.041285608902427086	4567.14820934

# ST\_

Cuando necesitas funciones que solo existen para geometry:

```
CREATE TABLE incidentes_tunjuelito_geog AS
SELECT
  ST_Transform(geom, 4326)::geography AS geog,
  id,
  -- agrega aquí otras columnas que quieras conservar
  servicio,
  fecha
FROM "Incidentes_2024_Tunjuelito_4686_2";
```

```
SELECT * FROM incidentes_tunjuelito_geog LIMIT 5;
```

	 geog geography		id integer		servicio character varying (254)		fecha date	
1	01010000A0E610000065339EB3A48852C0BB583E86656012400000000000000000		1		11. ATENCIÓN PREHOSPITALARIA - APH		2024-08-09	
2	01010000A0E610000003D6306F4C58752C0215FFB657A381240000000000000000		2		22. EMERGENCIA SIN INTERVENCION		2024-08-31	
3	01010000A0E61000000B1E77F24C98752C09DF92A4CF7311240000000000000000		3		7. INCIDENTES CON ÁRBOLES		2024-08-26	
4	01010000A0E6100000098856535998852C0D0D4A8DB62541240000000000000000		4		2. MATPEL		2024-08-24	
5	01010000A0E610000007BA7FB80218952C08F9AFA8CD7511240000000000000000		5		20. FALSA ALARMA		2024-08-07	

# ¿Cuándo usar geography vs geometry?

Escenario	geometry	geography
Datos locales	✓	✗
Datos globales	✗	✓
Funciones espaciales amplias	✓	✗
Medición precisa	✗	✓



# funciones nativas

## 1. Conversión entre formatos

- `ST_AsText(geography) → text`: Convierte un valor geography a texto en formato WKT (Well-Known Text).
- `ST_GeographyFromText(text) → geography`: Convierte un texto WKT a un tipo geography.
- `ST_AsBinary(geography) → bytea`: Devuelve la representación binaria WKB (Well-Known Binary) de una geography.
- `ST_GeogFromWKB(bytea) → geography`: Convierte WKB (bytea) a un tipo geography.
- `ST_AsSVG(geography) → text` : Representa la geography como un string en formato SVG. Útil para visualización web
- `ST_AsGML(geography) → text`: Convierte la geography a GML (Geography Markup Language).



# funciones nativas

## 1. Conversión entre formatos

- `ST_AsKML(geography) → text`: Convierte la geography a KML (Keyhole Markup Language), usado en Google Earth.
- `ST_AsGeoJson(geography) → text`: Convierte una geography a GeoJSON (muy usado en web y APIs).

## 2. Medición y análisis espacial

- `ST_Distance(geography, geography) → double`: Calcula la distancia (en metros) entre dos puntos geography
- `ST_AsGeoJson(geography) → text`: Convierte una geography a GeoJSON (muy usado en web y APIs).





# funciones nativas

## 2. Medición y análisis espacial

- `ST_DWithin(geometry, geometry, float8)` → boolean: Devuelve true si las dos geometrías están dentro de la distancia dada (en metros).
- `ST_Area(geometry)` → double: Devuelve el área en metros cuadrados de una superficie.
- `ST_Length(geometry)` → double: Devuelve la longitud de una línea en metros



# funciones nativas

## 3. Relaciones espaciales

- `ST_Covers(geography, geography)` → boolean: Devuelve true si la primera geography cubre completamente a la segunda.
- `ST_CoveredBy(geography, geography)` → boolean: Devuelve true si la primera geography está completamente contenida dentro de la segunda
- `ST_Intersects(geography, geography)` → boolean: Devuelve true si las dos geography se cruzan o tocan.

## 4. Geometrías derivadas

- `ST_Buffer(geography, float8)` → geography: Crea un área circular (buffer) en metros alrededor de una geography.
- `ST_Intersection(geography, geography)` → geography: Devuelve la geometría común (intersección) entre dos geography.



# GRACIAS

Cristian Delgado