- **Nota para el editor: por favor agregar los links que están en las funciones que en su nombre están en azul**

Capítulo 28: Geometrías en 3-D por Juan Pablo Grillo Torres

¿Qué son las geometrías en 3-D?

A lo largo de los capítulos anteriores se ha ido abordando el manejo de geometrías, sus funciones e integraciones, pero esto implementando geometrías en 2-D las cuales están conformadas por las coordenadas X, Y, en este capítulo se abordan las geometrías en 3-D y 4-D. En la extensión de PostGIS soporta las geometrías en 3 y 4 dimensiones las cuales están clasificadas por las siguientes dimensiones:

- Dimensión Z: esta dimensión agrega la información de altura.
- Dimensión M: esta dimensión tiene múltiples usos dentro de las geometrías, sus representaciones dependen del uso, unos ejemplos que se encuentran del uso de estos son el tiempo, milla de carretera o la información de distancia de subida.

Cuando se implementan estas dimensiones a las coordenadas X e Y se generan las geometrías en 3-D si se añade una de las dimensiones antes mencionadas o 4-D si se añaden ambas. Existen diversos software encargados del manejo de este tipo de archivos como son Cesium, ArcGIS, QGIS y en este caso se manejara la información usando la extensión de PostGIS.

¿Cómo se agregan en PostGIS?

Anteriormente, cuando se representaba una geometría en formato WKT, se representaba la geometría con su nombre seguido de la representación de los puntos o vértices de la geometría con un espacio de por medio mientras que para representar geometrías en 3-D y 4-D es necesario representar las geometrías correctamente en las que seguido al nombre de la geometría se añade la letra de la dimensión respectiva que se describe y además en las representaciones de las coordenadas se incluye la coordenada adicional para cada punto o vértice de la geometría.

	Representación en 2 dimensiones	Representación con Z	Representación con M	Representación con Z y M
Punto	Point (0 0)	Point Z(0 0 0)	Point M(0 0 0)	Point ZM(0 0 0 0)
Línea	Linestring(0 0,1 1)	Linestring Z(0 0 0,1 1 1)	Linestring M(0 0 0,1 1 1)	Linestring ZM(0 0 0 0,1 1 1 1)
polígono	Polygon(0 0,1 1,1 0)	Polygon Z(0 0 0,1 1 1, 1 1 0)	Polygon M(0 0 0,1 1 1, 1 1 0)	Polygon ZM(0 0 0 1,1 1 1 1, 1 1 0 1)

Es importante tener en cuenta algunas recomendaciones cuando se hace el manejo de geometrías en un número mayor a 2 dimensiones de las cuales encontramos

- Tener una correcta sintaxis de los datos en formato WKT y WKB,
- la dimensión adicional que se menciona en el formato WKT o con las funciones que pueden forzar datos 2 a 3-D o 4-D o viceversa estén correctamente especificadas a qué dimensión se reduce.
- A pesar de que en la representación de los datos en WKT sus datos estén solamente con ceros en la tercera o cuarta coordenada no significa de que estén los datos en 2 dimensiones.

Otras geometrías usadas más avanzadas y la notación BNF

Adicionalmente en otro capítulo se han mencionado otras geometrías adicionales además de las ya manejadas usualmente que son la línea, el punto y el polígono, las cuales son capaces de representar otros objetos espaciales que se pueden representar de los que encontramos.

- Curva (curve)
- Cuerda circular (circularstring)
- Curva compuesta (compoundcurve)
- Superficie (surface)
- Curva poligonal (curvepolygon)
- triángulo(triangle)
- Colección (Collection)
- Superficie poliédrica (polyhedricalsurface)
- Red de triángulos irregulares (TIN)
- Colección de geometrías (geometrycollection)

De los cuales todos estos elementos también pueden ser representados en 3 o 4 dimensiones o algunos no existe su representación en 2 Dimensiones como sucede con el caso de la red de triángulos irregulares y las superficies poliédricas de las cuales su representación está formada por 3 dimensiones usando la coordenada Z para

representar la altura de estas geometrías, tienen su respectiva representación binaria según la notación backups Naur Form (BNF) en la que las geometrías se pueden representar de múltiples formas en binario e incluso representándose como lo harían otras geometrías, los únicos casos en los que la representación BNFde la geometría no tiene forma alternativa para representarse es con los puntos, curvas, líneas, superficies y polígonos.

Uso de los archivos de ejemplo, convertir geometrías de 2-D a 3-D

Como principalmente en Colombia se hace el manejo de las geometrías en 2-D en este capítulo se introduce el código para la base de datos de la localidad de Suba que incluye las siguientes capas:

- Barrios de suba
- Localidad de suba
- Edificios
- Multas de tránsito
- Vías de Suba

En este caso la capa más relevante para su uso en 3-D va a ser la capa de edificios en la que a esta capa de multipoligonos se añadirá primero las coordenadas Z y M en una nueva columna como mecanismo de Backup conservándose la columna de geometrías original para posteriormente agregarse las alturas a los polígonos deseados según el criterio deseado.

Primero se genera la geometria en 4-D mediante la siguiente función:

```
alter table
   edificios
   ADD COLUMN
      geom_4d geometry(MultiPolygonZM, 4326);
```

Si se deseara añadir una columna en 3-D es necesario especificarla en su nombre, asi mismo el tipo de geometría que se representa y el SRID que maneja la tabla a modificar.

Posteriormente se hace la inserción de los valores de las coordenadas en 2-D en la columna en 3-D mediante la función st_Force4D.

```
UPDATE edificios
        SET geom_4d = ST_Force4D(geom)
```

En el caso de que se usen geometrías en 3D se implementa la función st_Force3D y si

es en 3-D pero usando la dimensión M se usa la función St_force3DM.

Una vez agregada la columna de geometrías en 4D en esta caso verificamos que se haya adicionado correctamente las coordenadas de un poligono, eto haciendo selección de una geometría 4-D y limitando la salida a apenas un resultado.

```
      select ST_AsText(geom_4d)
 from edificios
limit 1
```

	st_astext text
1	MULTIPOLYGON ZM (((-74.055512 4.733596 0 0,-74.0554991 4.7336819 0 0,-74.0553861 4.733665 0 0,-74.055399 4.7335791 0 0,-74.055512 4.733596 0 0)))

Como se puede observar la columna fue creada correctamente y en este caso también se conserva la columna geom en caso de que se genere un error agregando las coordenadas u modificando los valores,

Se puede observar de que los valores de las coordenadas Z y M se encuentran completamente con valores de 0 por lo que a partir de este punto se deben crear las condiciones lógicas para modificar la coordenada Zmediante la función ST_Translate la cúal permite que se puedan agregar coordenadas a las coordenadas actuales de una o varias geometrías, es importante destacar que que la función ST_Translate solo permite hacer modificación de las coordenadas X, Y y Z por lo que no es posible que se realice una modificación de la coordenada M. En este caso se va a hacer una modificación de la altura de los barrios ubicados dentro del barrio de Mazuren en la localidad de Suba mediante el uso de un Update con la siguiente estructura.

```
UPDATE
        edificios
        SET geom_4d = ST_Translate(edificios.geom_4d,0,0,10)
                where EXISTS(
SELECT geom_4d
        FROM barrios_suba AS suba
                WHERE suba.fid = 87
        AND ST_Within(edificios.geom, suba.geom))
```

Posteriormente se hace una función de selección para verificar que se modificaron correctamente las alturas de los edificios en el barrio Mazuren con la siguiente función.

```
select ST_AsText(geom_4d)
        FROM edificios
                        JOIN barrios_Suba AS suba
                        ON st_within(edificios.geom,suba.geom)
```

```
        WHERE suba.fid = 87
        limit 1
```

	st_astext text
1	MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-

Como se puede observar se ha añadido correctamente en este caso la altura para los edificios que están contenidos dentro del barrio Mazuren

Adicionalmente para describir de mejor forma el comportamiento de las funciones en 3-D se va a añadir una altura distinta a los edificios de Victoria los cuales están contenidos en el barrio que está al sur de Mazuren mediante la siguiente función de actualizacion de tablas.

```
UPDATE
        edificios
        SET geom_4d = ST_Translate(edificios.geom_4d,0,0,30)
                where EXISTS(
SELECT geom_4d
        FROM barrios_suba AS suba
                WHERE suba.fid = 61
        AND ST_Within(edificios.geom, suba.geom))
```

Por ultimo se hace la verificacion de las ultimas modificaciones con 2 objetivos de los cuales son el de verificar que la adicion de altura se haya realizado correctamente y de que en la sintaxis de las funciones que usan la funcion St_translate no haya afectado de forma no deseada a otros edificios generando que el trabajo actual haya sido en vano.

```
select ST_AsText(geom_4d)
        FROM edificios
                        JOIN barrios_Suba AS suba
                        ON st_within(edificios.geom,suba.geom)
                WHERE suba.fid = 61
                limit 1
```

	st_astext text
1	MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-

En este caso los resultados son coherentes a las actualizaciones realizadas por lo que se puede continuar con las funciones en 3-D,

Funciones implementadas para geometrías en 3-D

**Nota: En este punto es importante destacar que el SRID que maneja las coordenadas de las columnas Geom y Geom_4D son manejadas con el elipsoide WGS 84 por lo que los resultados vistos serán en grados decimales, por lo tanto los resultados que se visualizaran tendrán una diferencia notable, es importante hacer estas modificaciones de altura usando SRIDs que implementen medidas en metros para poder obtener resultados realistas y precisos a la realidad, pero para la descripción de datos en este capítulo ayudan a distinguir las diferencias entre las funciones en 2-D que se han ido manejando en capítulos anteriores por lo que a forma de comparación se usarán los 2 mismos polígonos de comparación para todas las funciones tanto en su forma 3-D como 2-D. Por otra parte la visualización gráfica en 3-D de las geometrías seleccionadas no es posible de realizar en PostGIS, su visualización terminara en que no se visualice ningún elemento e incluso es posible que para posteriores consultas en 2-D usando la misma sesión no se lleguen a visualizar, posiblemente por algún error inducido por la visualización de las geometrías en 3-D.*

Con las funciones en 3D que se van a implementar se han trabajado en capítulos anteriores una o varias veces, pero en este caso se implementa su versión en 3-D la cual para los cálculos realizados por las funciones requiere que se tenga la dimensión Z en su estructura e implementando los siguientes polígonos extraídos de las consultas anteriores.

Polígono de Mazuren:

MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))

Polígono de Victoria norte:

MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))

- ST_3DClosestPoint(geometría 1,geometría 2): esta función retorna las coordenadas X,Y,Z del punto en el que la geometría 1 está más cerca de la geometría 2, si se da el caso de que tanto la geometría 1 como la 2 están en 2-D retorna los valores de las coordenadas en 2-D de igual forma como si se usara

la función ST_ClosestPoint.

Su implementación en funcion es la siguiente asi como su resultado

```
SELECT
        ST_AsEWKT(ST_3DClosestPoint(line,pt)) AS cp3d_line_pt,
      ST_AsEWKT(ST_ClosestPoint(line,pt)) As cp2d_line_pt
   FROM (SELECT 'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))'::geometry As pt,
          'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))  '::geometry As line
      )
```

	cp3d_line_pt text	cp2d_line_pt text
1	POINT(-74.0555897 4.7327995 30)	POINT(-74.0555897 4.7327995)

- ST_3DDistance(geometría 1, geometría 2): retorna la mínima distancia cartesiana tridimensional entre dos geometrías en unidades proyectadas, si se da el caso de que ambas geometrías están en 2-D su funcionamiento es el mismo al de la función ST_Distance, es importante tener en cuenta de que para que los resultados sean coherentes a la realidad el SRID de ambas geometrías tiene que ser el mismo.

```
SELECT
      ST_3DDistance(geom1,geom2) AS Distance3d,
      ST_Distance(geom1,geom2) As Distance2d
FROM (
   SELECT
       'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))'::geometry AS geom1,
       'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30
```

0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))'::geometry AS geom2
)

	distance3d double precision	distance2d double precision
1	20.000000905949936	0.006019800512479453

- ST_3DDWithin(geometría 1,geometria 2, Distancia) retorna verdadero si la distancia entre dos objetos espaciales está dentro de la distancia medida en las unidades que use el SRID de ambas geometrías, si no hay coordenadas Z presentes se toma la función como si fuera la función ST_DWithin().

```
SELECT
        ST_3DDWithin(geom1,geom2,20) AS Within3d,
      ST_DWithin(geom1,geom2,20) As Within2d
FROM (
   SELECT
        'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))'::geometry AS geom1,
        'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))'::geometry AS geom2
)
```

	within3d boolean	within2d boolean
1	false	true

Al aplicar este caso se observa un comportamiento de las geometrias de multipoligonos con altura en la que bajo estas circunstancias deberia de mostrar un valor verdadero si se tomara el poligono como un solido desde su base, pero en este caso los calculos de intersección se toman como si cada polígono fuera una hoja de papel a una determinada altura, por lo que a pesar de hacer el ST_3DDWithin con una distancia de 20 grados decimales no se intersectan los 2 poligonos, mientras que en su versión en 2-D si lo hacen.

- ST_3DDFullyWithin(geometría 1,geometría 2, distancia): retorna verdadero si ambas geometrías se encuentran completamente contenidas a una distancia especificada, se tiene en cuenta el SRID de las geometrías, por lo tanto, debe de ser el mismo para tener coherencia en sus resultados, su función homóloga en dos dimensiones es ST_DFullyWithin() , por lo tanto funciona igual a esta si las coordenadas de ambas geometrías están en 2 dimensiones.

```
SELECT
      ST_3DDFullyWithin(geom1,geom2,20) AS FullyWithin3d,
    ST_DFullyWithin(geom1,geom2,20) As FullyWithin2d
FROM (
  SELECT
      'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))'::geometry AS geom1,
      'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))'::geometry AS geom2
) AS subquery;
```

	fullywithin3d boolean	fullywithin2d boolean
1	false	true

- ST_3DIntersects(geometría 1,geometría 2): retorna verdadero si las geometrías intersecan espacialmente en uno o más puntos, esta relación se realiza en 3 dimensiones, solo puede ser aplicada para puntos y líneas.

```
SELECT
      ST_3Dintersects(geom1,geom2) AS intersects3d,
      ST_intersects(geom1,geom2) As intersects2d
FROM (
  SELECT
      'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599
```

4.7369455 10 0)))'::geometry AS geom1,
'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0)))'::geometry AS geom2
)

	intersects3d boolean	intersects2d boolean
1	false	false

Debido a que los dos edificios están ubicados en espacios distintos en este caso no se intersectan en 2-D ni en 3-D

- ST_3DLongestLine(geometría 1,geometría 2): retorna la línea más larga entre dos geometrías, si hay dos o más líneas que cumplen esta condición retorna solo una línea de distancia, esta función es equivalente a usar la función ST_3DMaxDistance() y su función homologa en geometrías en 2-D es ST_LongestLine().

SELECT
ST_AsEWKT(ST_3dLongestLine(geom1,geom2)) AS LongestLine3d,
ST_AsEWKT(ST_LongestLine(geom1,geom2)) As LongestLine
FROM (SELECT 'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599 4.7369455 10 0)))'::geometry As geom1,
'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906 30 0))) '::geometry As geom2
)

	longestline3d text	longestline text
1	LINESTRING(-74.0496976 4.7368707 10,-74.0556915 4.7326011 30)	LINESTRING(-74.0496976 4.7368707,-74.0556915 4.7326011)

- ST_3DMaxDistance(geometría 1,geometría 2): como se mencionó anteriormente retorna la línea más larga que hay entre dos geometrías, pero en este caso se considera el SRID y se da el resultado en unidades proyectadas, es útil cuando se utiliza un SRID que implementa coordenadas en grados decimales y se desea conocer su resultado en metros, su función homologa en 2'D es ST_MaxDistance().

```
SELECT
        ST_3DMaxDistance(geom1,geom2) AS MaxDistance3d,
      ST_MaxDistance(geom1,geom2) As MaxDistance2d
FROM (
   SELECT
        'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976 4.7368707 10
0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10 0,-74.0498436 4.735654 10
0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10 0,-74.0496673 4.7353341
10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727 10 0,-74.0499767
4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221 4.7351625 10
0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10 0,-74.0500135 4.7354094
10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334 10 0,-74.0499599
4.7369455 10 0)))'::geometry AS geom1,
        'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011 30
0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206 4.7325906
30 0)))'::geometry AS geom2
)
```

	maxdistance3d double precision	maxdistance2d double precision
1	20.00000135390799	0.007359097863862985

- ST_3DShortestLine(geometría 1,geometría 2): de forma contraria a la función ST_3DLongestLine esta función retorna la línea más corta presente entre dos geometrías, si se presentan dos o más casos en la que hay una línea más corta se devuelve solo un resultado y su funcionalidad es igual a la función ST_3DDistance() pero sin tener en cuenta que el resultado devuelto está en unidades proyectadas.

```
SELECT
      ST_AsEWKT(ST_3dShortestLine(geom1,geom2))as shortestline3d,
      ST_AsEWKT(ST_ShortestLine(geom1,geom2)) as shortestline
   FROM (SELECT 'MULTIPOLYGON ZM (((-74.0499599 4.7369455 10 0,-74.0496976
4.7368707 10 0,-74.0498604 4.7357673 10 0,-74.0498792 4.7356651 10
0,-74.0498436 4.735654 10 0,-74.0497576 4.7355331 10 0,-74.0496569 4.7353914 10
0,-74.0496673 4.7353341 10 0,-74.0497032 4.7350678 10 0,-74.0498694 4.7349727
10 0,-74.0499767 4.7349897 10 0,-74.0499553 4.7351385 10 0,-74.0499221
4.7351625 10 0,-74.0499083 4.7352552 10 0,-74.0499644 4.7353374 10
0,-74.0500135 4.7354094 10 0,-74.0501078 4.7354228 10 0,-74.0501767 4.7354334
10 0,-74.0499599 4.7369455 10 0)))'::geometry As geom1,
          'MULTIPOLYGON ZM (((-74.0556206 4.7325906 30 0,-74.0556915 4.7326011
30 0,-74.0556606 4.7328099 30 0,-74.0555897 4.7327995 30 0,-74.0556206
4.7325906 30 0)))   '::geometry As geom2
      )
```

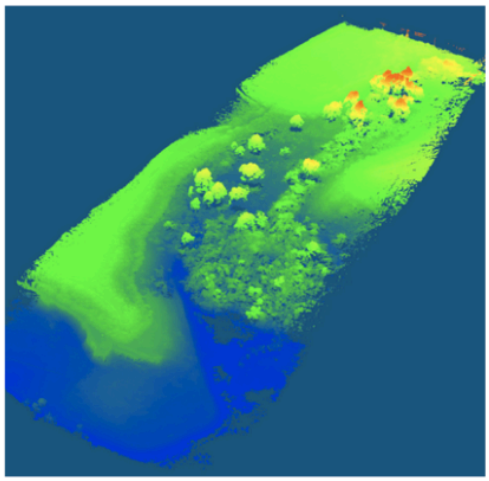
	shortestline3d text	shortestline text
1	LINESTRING(-74.0501767 4.7354334 10,-74.0555897 4.7327995 30)	LINESTRING(-74.0501767 4.7354334,-74.0555897 4.7327995)

Creación e implementación de índices espaciales en N-dimensiones

Es importante destacar de que estas funciones espaciales desde la versión 2.0.0 soportan superficies poliédricas así como triángulos y redes irregulares de triángulos (TIN), de igual forma a partir de esta versión las funciones soportan las funciones en 3-D y además no dejan de usar los índices Z que se explicaran más adelante.
Para el caso de los índices espaciales es importante conocer en que situaciones se deben de aplicar debido a que el objetivo de los índices espaciales es de optimizar la velocidad de la búsqueda de una consulta espacial, algunos ejemplos mencionados en los que se ejemplifica la eficiencia o falta de esta son:

- Buen candidato: Consulta con un set de trazas GPS
- Mal candidato: consulta usando una nube de puntos de un modelo digital de elevación (DEM)

Aplicación incorrecta



Aplicación correcta

Para la adicion de los indices epaciales n dimensionales es necesario especificar que son N-dimensionales, en este caso se genera uno implementando GIST de forma similar a como se ha creado en capitulos anteriores pero en este caso se especifica mediante la especificacion gist_geometry_ops_nd:

CREATE INDEX edificios_gix_nd ON edificios
USING GIST (geom_4d gist_geometry_ops_nd);

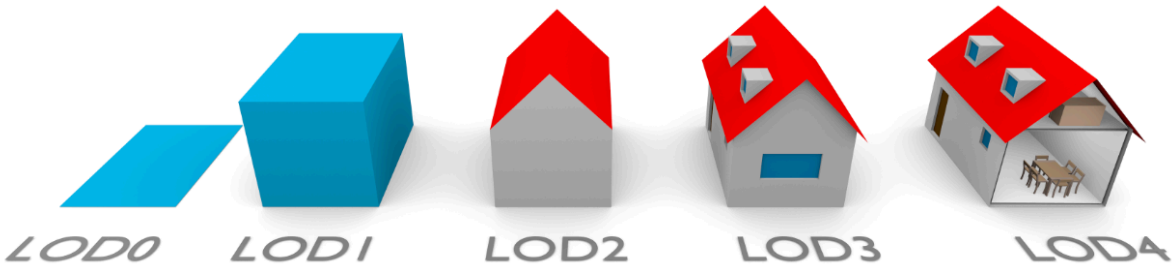
Una vez creado el indice espacial es posible implementarlo en las consultas en 3-D implementando el &&&, en vez de usar los simbolos && que se usa especificamente en 2 dimensiones. Un ejemplo de su uso se encuentra en la siguiente funcion la cual realiza un Bounding box en tres dimensiones.

```
SELECT *
FROM edificios
WHERE geom_4D &&&
            ST_SetSRID('MULTIPOLYGON(((-74.0556206 4.7325906,-74.0556915
4.7326011,-74.0556606 4.7328099,-74.0555897 4.7327995,-74.0556206
4.7325906)))'::geometry,9377);
```

Que es el modelo CityGML, ejemplos de su uso e importancia

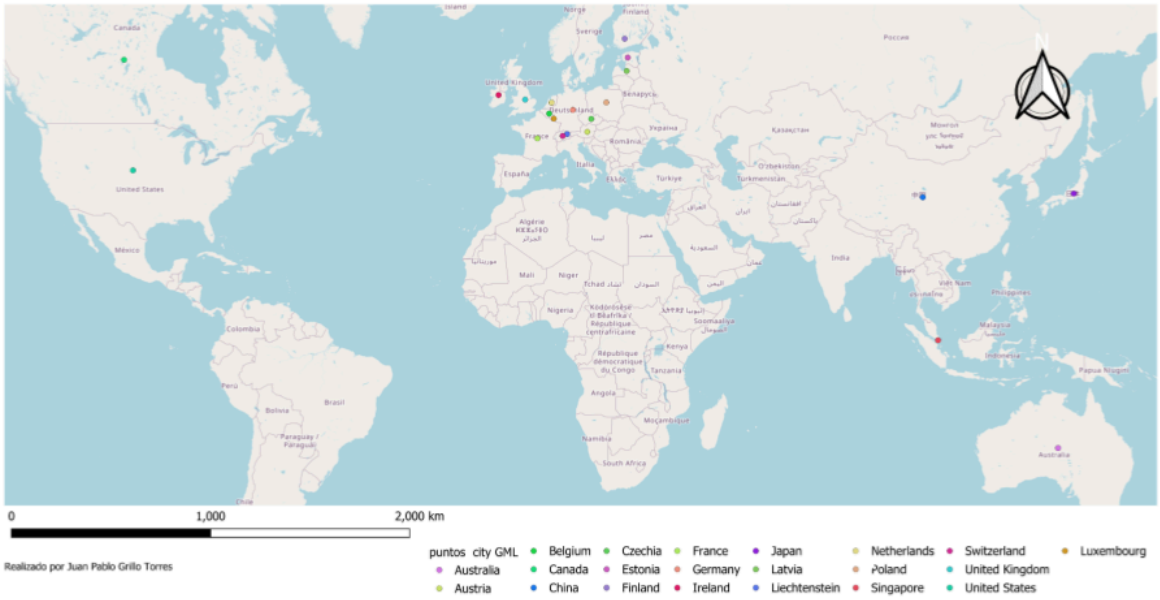
Adicionalmente al capítulo ya descrito se describe un tema relacionado con las geometrías en 3-D el cual es el modelo CityGML, este es un estándar reconocido por la OGC (Open Spatial Consortium) el cual describe un modelo de datos para su almacenaje e intercambio de modelos tridimensionales, este modelo está basado en el estándar GML y es implementado esencialmente para las ciudades y tiene diversas utilidades como los son su uso en actividades turísticas, diseño arquitectónico, catastro en 3-D, comunicaciones móviles, simulación con capacitación, aplicaciones energéticas, planeación urbanística y paisajística y entre otras.
En la versión más reciente del modelo, que es la versión 3.0 describe un modelo común de información semántica para la representación de objetos urbanos, enfocado principalmente a algunas tareas relacionadas con la I.A como los son los gemelos digitales y la conducción autónoma.
Los objetos dentro del modelo CityGML se clasifican de forma sencilla según su nivel de detalle, en esta hay 5 categorías distintas que describen de forma menos detallada a más detallada de los objetos, cada uno se clasifica mediante él nombre LOD seguido del número y se definen de la siguiente forma:

- LOD0: Representación de las huellas y los polígonos del techo acá se hace la transición de 2D a 3D.
- LOD 1: Modelo prismático que se genera mediante la extrusión del LOD0.
- LOD2: Es un modelo con una representación simple del techo del edificio u otros elementos semánticos.
- LOD3: Representa un modelo arquitectónico detallado que incluye puertas y ventanas.
- LOD4: Incluye todas las características de los modelos anteriores añadiendo características de los interiores.



Existen algunos proyectos de ciudades representadas en el modelo CityGML de los cuales uno de estos que es accesible para todo el público es el de Awesome CityGML, el cual consiste en la digitalización de diversa información como lo son las construcciones, espacio de la calle, vegetación, terreno, mobiliario urbano, texturas u otros objetos genéricos de varias ciudades de Europa, Estados Unidos, Canadá, Asia y Oceanía.

Países con ciudades representadas según el modelo CityGML



Cabe destacar que la representación de ciudades que cumplen el estándar de CityGML en latinoamérica y más específicamente en Colombia es demasiado baja, aun con todas las implementaciones que puede tener el uso de este estándar para diversas aplicaciones antes mencionadas.