Київський національний університет будівництва і архітектури
Кафедра геоінформатики і фотограмметрії

КВАЛІФІКАЦІЙНА РОБОТА

Дослідження засобів моделювання рельєфу в середовищі СКБД PostgreSQL/PostGIS

Виконав студент групи ГіСТм-23
ДУЛЬСЬКИЙ Володимир Сергійович

Керівник: проф., д.т.н. ЛЯЩЕНКО Анатолій Антонович

Київ - 2024

Актуальність та мета робити

Актуальність теми. Моделювання рельєфу є ключовим аспектом багатьох сфер діяльності, таких як ГІС, міське планування та екологічний моніторинг. Сучасні ГІС забезпечують потужні засоби для створення цифрових моделей рельєфу (ЦМР), проте зростає тенденція до інтеграції баз геоданих з універсальними СКБД, що дозволяє працювати з векторними, растровими даними та ЦМР в одному середовищі. Це зменшує залежність від специфічного інструментарію ГІС і форматів даних. Об'єктно-реляційна СКБД PostgreSQL з просторовим розширенням PostGIS є однією з найпоширеніших СКБД з відкритим кодом, що використовується в сучасних ГІС. PostgreSQL/PostGIS також відповідає міжнародним стандартам і специфікаціям у сфері географічної інформації.

Метою роботи є дослідження функціональних можливостей та ефективності використання СКБД PostgreSQL/PostGIS для створення і використання цифрових моделей рельєфу на реальних обсягах даних при вирішенні типових прикладних задач.

Завдання дослідної роботи

1. аналіз стану і тенденцій розвитку засобів цифрового моделювання рельєфу в ГІС та СКБД;
2. аналіз уніфікованих базових типів геопросторових даних і функцій, що використовуються для моделювання рельєфу в базах геопросторових даних;
3. проведення обчислювального експеременту зі створення і використання GRID та TIN моделей рельєфу в середовищі СКБД PosrgreSQL/PostGIS;
4. розроблення прикладних SQL функцій для побудови 3D ліній з використанням GRID моделі рельєфу;
5. розроблення прикладних SQL функцій для побудови та реконструкції 3D трикутників TIN моделі рельєфу;
6. розроблення прикладних SQL функцій для інтерполяції висоти довільної 2D точки та побудови 3D ліній за TIN моделлю рельєфу.

Види ЦМР

Цифрова модель рельєфу (ЦМР) в загальному сенсі у геоінформатиці визначається як цифрове уявлення рельєфу земної поверхні, створене на основі даних про рельєф та морфології місцевості.

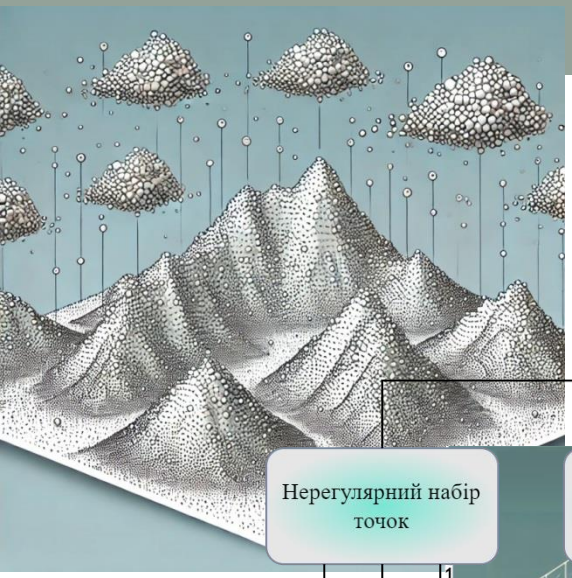
В англомовній літературі розрізняють такі формулювання ЦМР:

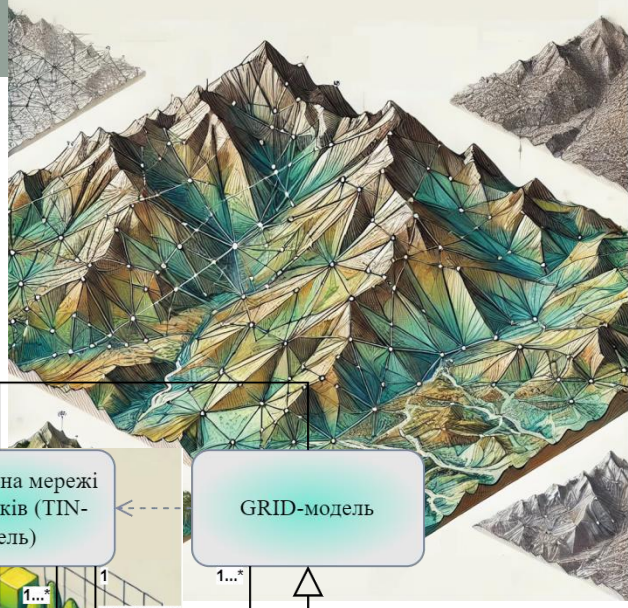
цифрова модель висот (digital elevation model, DEM) – це цифрове представлення поверхні землі, яке відображає лише голий ґрунт, виключаючи дерева, будівлі та інші поверхневі об'єкти;

цифрова модель рельєфу (digital terrain model, DTM) – містить інформацію про висоти тієї чи іншої місцевості без відображення об'єктів та зелених насаджень;

цифрова модель поверхні (digital surface model, DSM) – поняття з’явилось з появою фотограмметричних станціях, являє тривимірним представлення поверхні Землі, включаючи всі об'єкти та об'єкти, присутні на ній.

Типи ЦМР





Модель рельєфу

Нерегулярний набір точок

Набір структурних ліній

Цифрова картографічна модель (горизонталі)

Нерегулярна мережі трикутників (TIN-модель)

GRID-модель

1 вхідні дані

1 вхідні дані

1 вхідні дані

1...* вхідні дані

1 вхідні дані

1...* результат

1...* результат

1 вхідні дані

1...*

вхідні дані

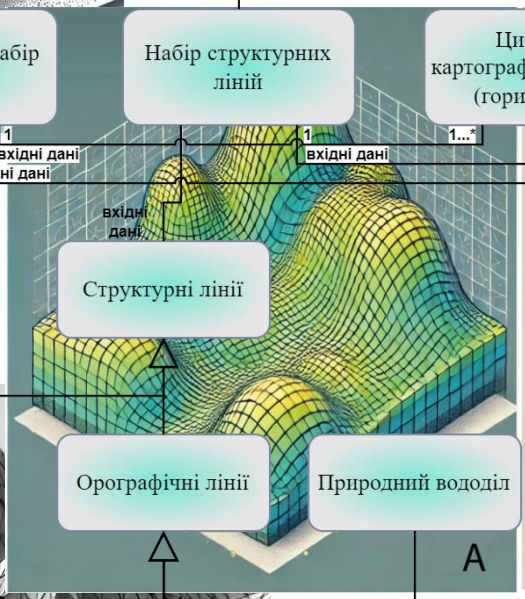
вхідні дані

Позначки висот (точки)

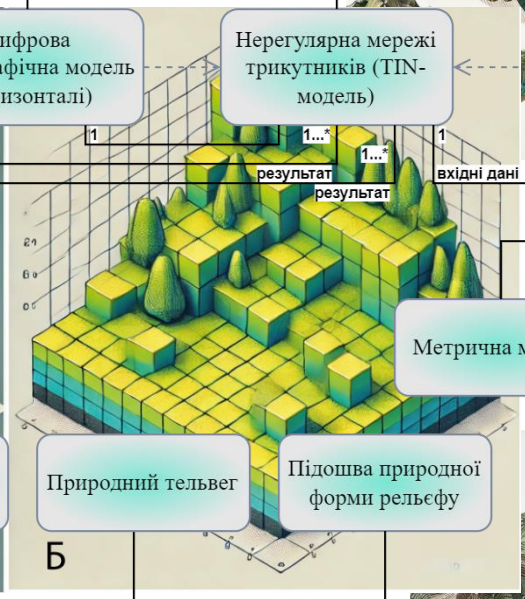
Структурні лінії

Метрична модель

Растрова модель



А



Б

Лінії шельфу змін рельєфу

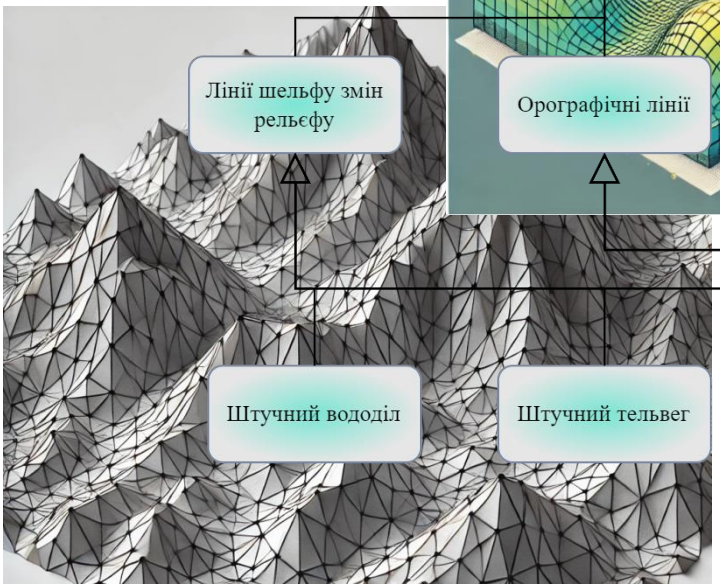
Орографічні лінії

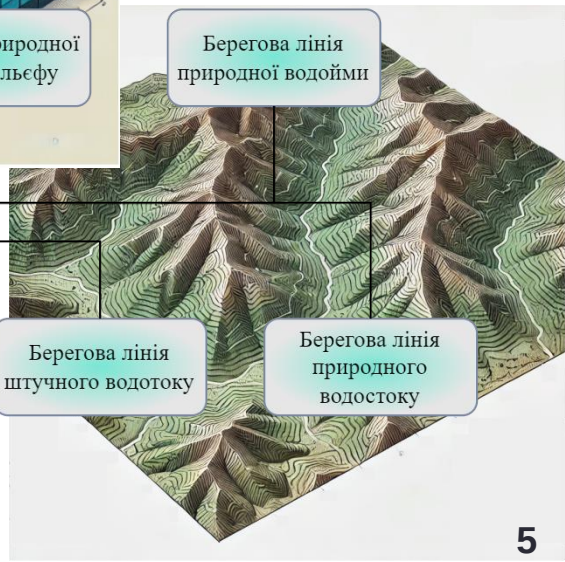
Природний вододіл

Природний тельвег

Підошва природної форми рельєфу

Берегова лінія природної водойми





Штучний вододіл

Штучний тельвег

Підошва штучної форми рельєфу

Берегова лінія штучної форми рельєфу

Берегова лінія штучного водотоку

Берегова лінія природного водостоку

Програмне забезпечення роботи з ЦМР

Функції	ArcGIS	MapInfo	Autodesk Map 3D	GeoMedia	QGIS	PostgreSQL
Підтримка векторної моделі із Z-координатою	+	-	+	+	+	+
Підтримка регулярної моделі	+	+	+	+	+	+
Підтримка тріангуляційної моделі	+	+	+	-	+	+
Введення даних із геодезичних приладів	+	-	+	-	+	+
Фототріангуляція	+	-	+	+	+	-
Автоматична векторизація	+	+	+	+	+	-
Тривимірна візуалізація	+	+	+	-	+	+
Інтерполяція висот	+	+	-	+	+	+
Побудова профілів	+	+	+	+	+	+
Побудова ізоліній	+	+	-	+	+	+
Розрахунок експозиції схилів	+	+	-	-	+	+
Розрахунок об’ємів земляних робіт	-	+	+	-	+	+
Аналіз видимості	+	+	+	-	+	+
Побудова тальвегів та вододілів	+	-	+	-	+	+

Загальні стандарти геоінформаційного моделювання

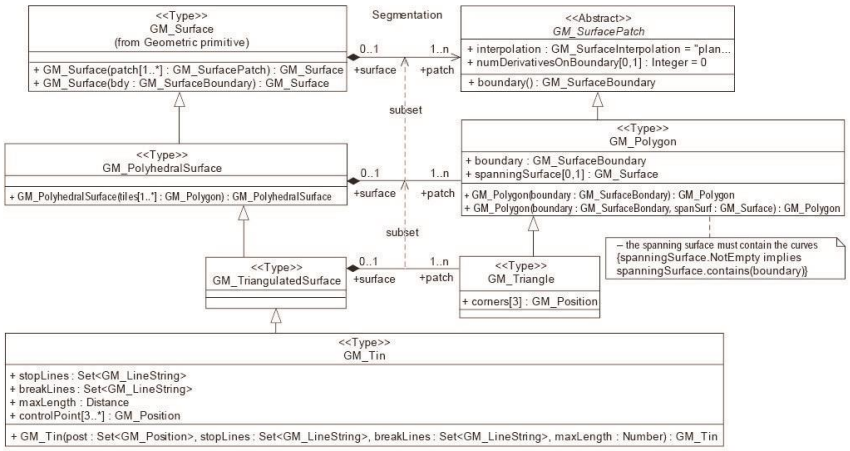
1. ISO 19107 Географічна інформація — Просторові схеми;
2. ISO 19125 Географічна інформація — Простий доступ до геопросторових об’єктів:

- ISO 19125-1: Частина 1 – Загальна архітектура;
- ISO 19125-2: Частина 2 – Опція SQL;

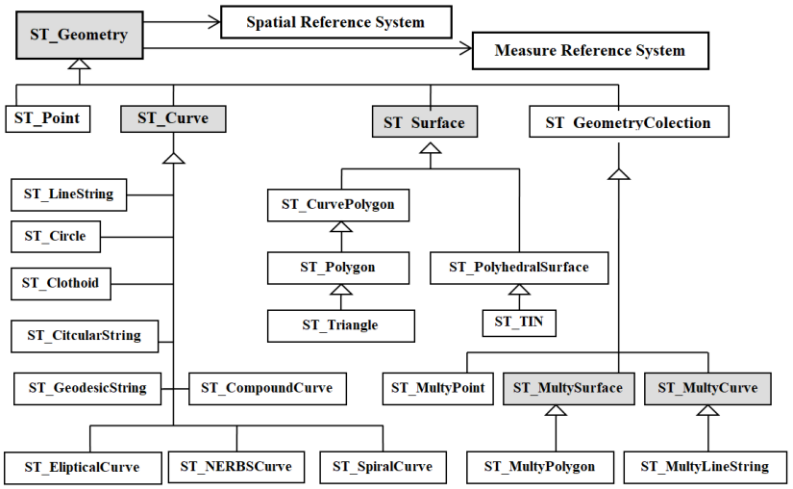
3. ISO 19123 Географічна інформація — Схема для геометрії та функцій покриттів;
4. Open Geospatial Consortium (OGC) Simple Feature Access;
5. ISO/IEC 13249-3 Інформаційні технології – Мови баз даних – SQL мультимедійні та прикладні пакети, Частина 3: Просторові дані;

Концептуальна модель класів об’єктів покриття та ієрархія класів об'єктів за стандартами ISO та OGC

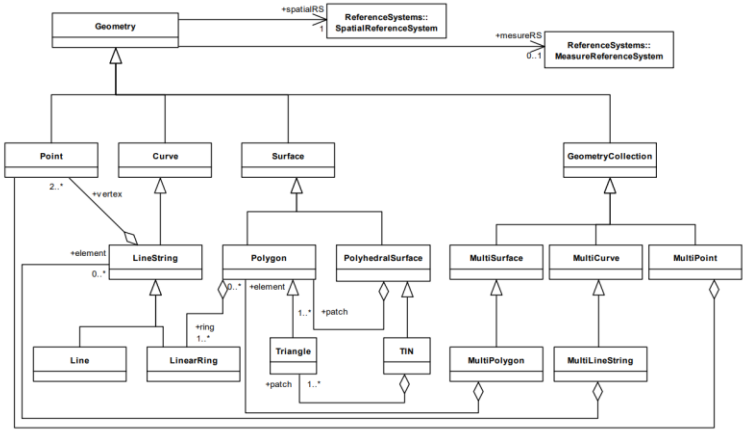
Концептуальна модель подання класів об’єктів для покриття за ISO 19123



Ієрархія класів об’єктів за ISO/IEC 13249-3



Ієрархія класів об’єктів за ISO 19125 та OGC



Типи даних для ЦМР в PostGIS

GRID:
raster

GRID-модель рельєфу в PostGIS реалізована на основі типу даних ***raster***, де растрові дані зберігаються у вигляді двовимірної матриці та розбиваються на блоки (тайли) для підвищення продуктивності і зручного доступу до окремих частин даних.

TIN:
Triangle/ TIN

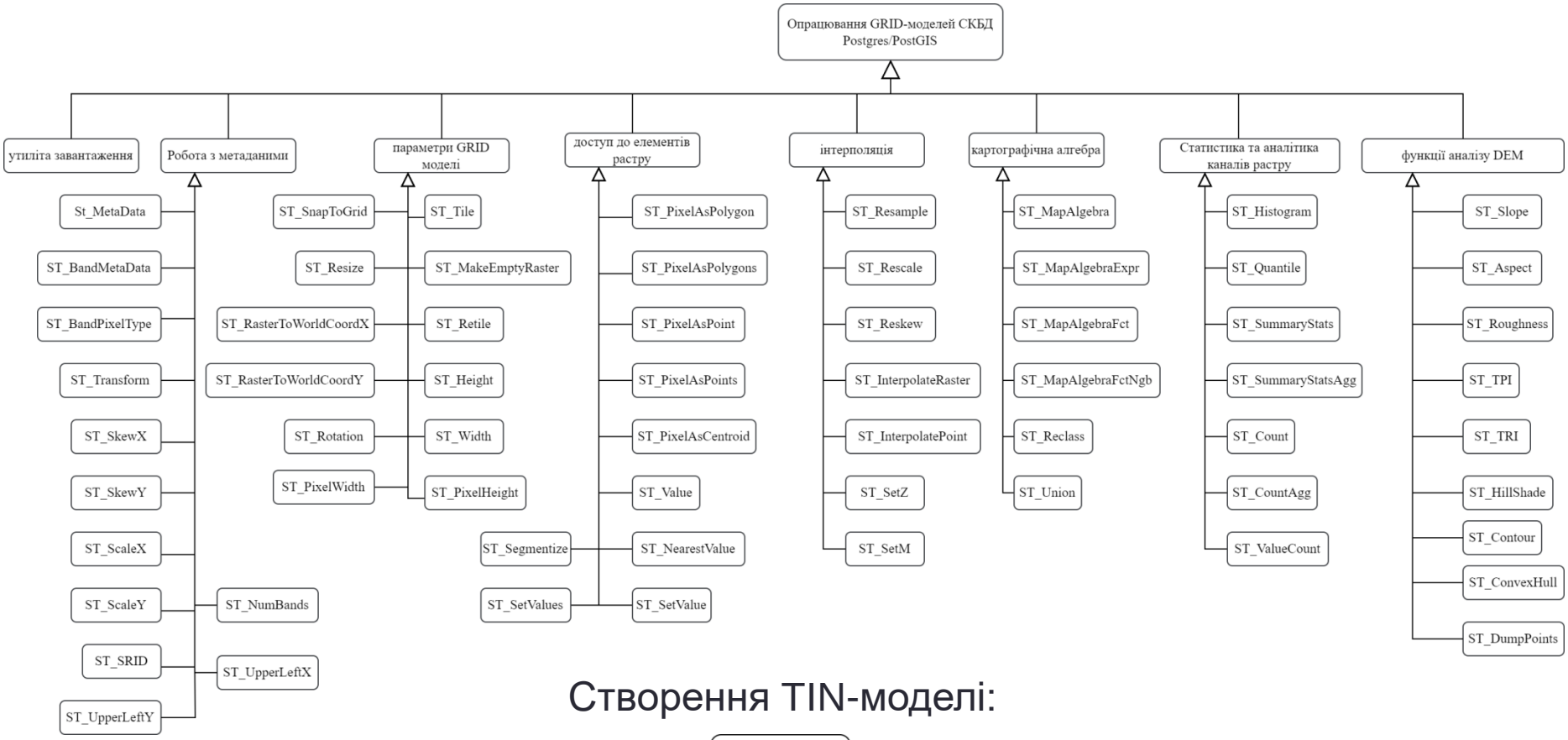
TIN подаэться типом даних ***Triangle*** та ***TIN.***

Triangle визначає трикутник як полігон з трьома неколінеарними вершинами.

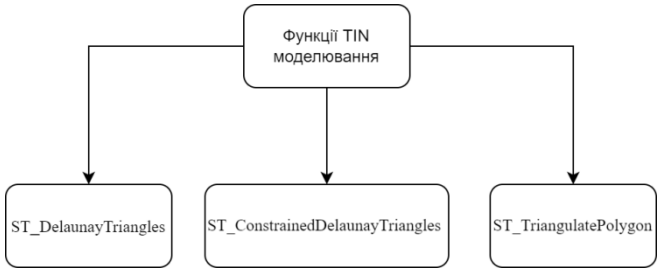
TIN є колекцією суміжних трикутників, які моделюють тривимірну нерегулярну поверхню покриття.

Функції опрацювання ЦМР в PostGIS

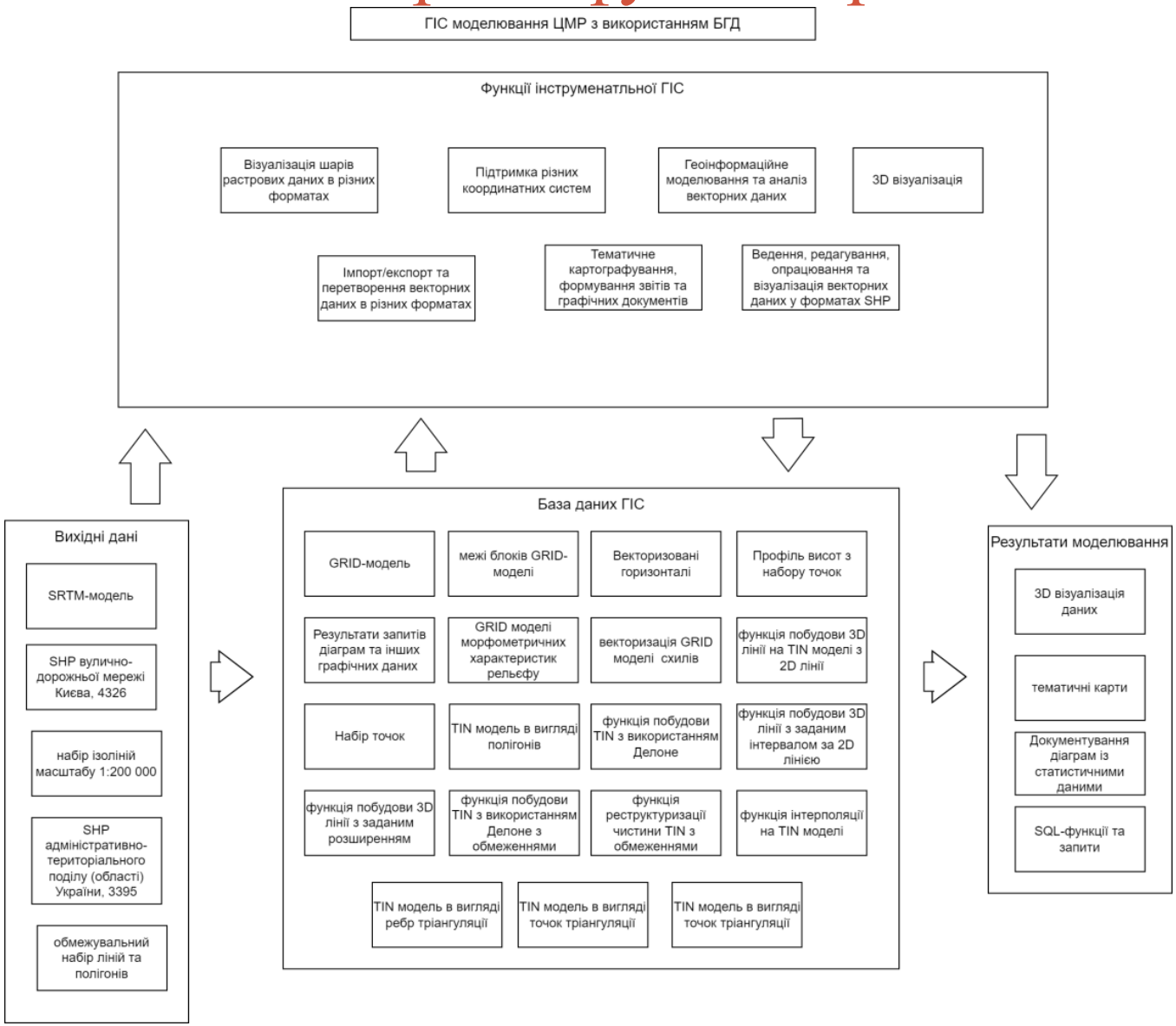
Опрацювання GRID-моделі:



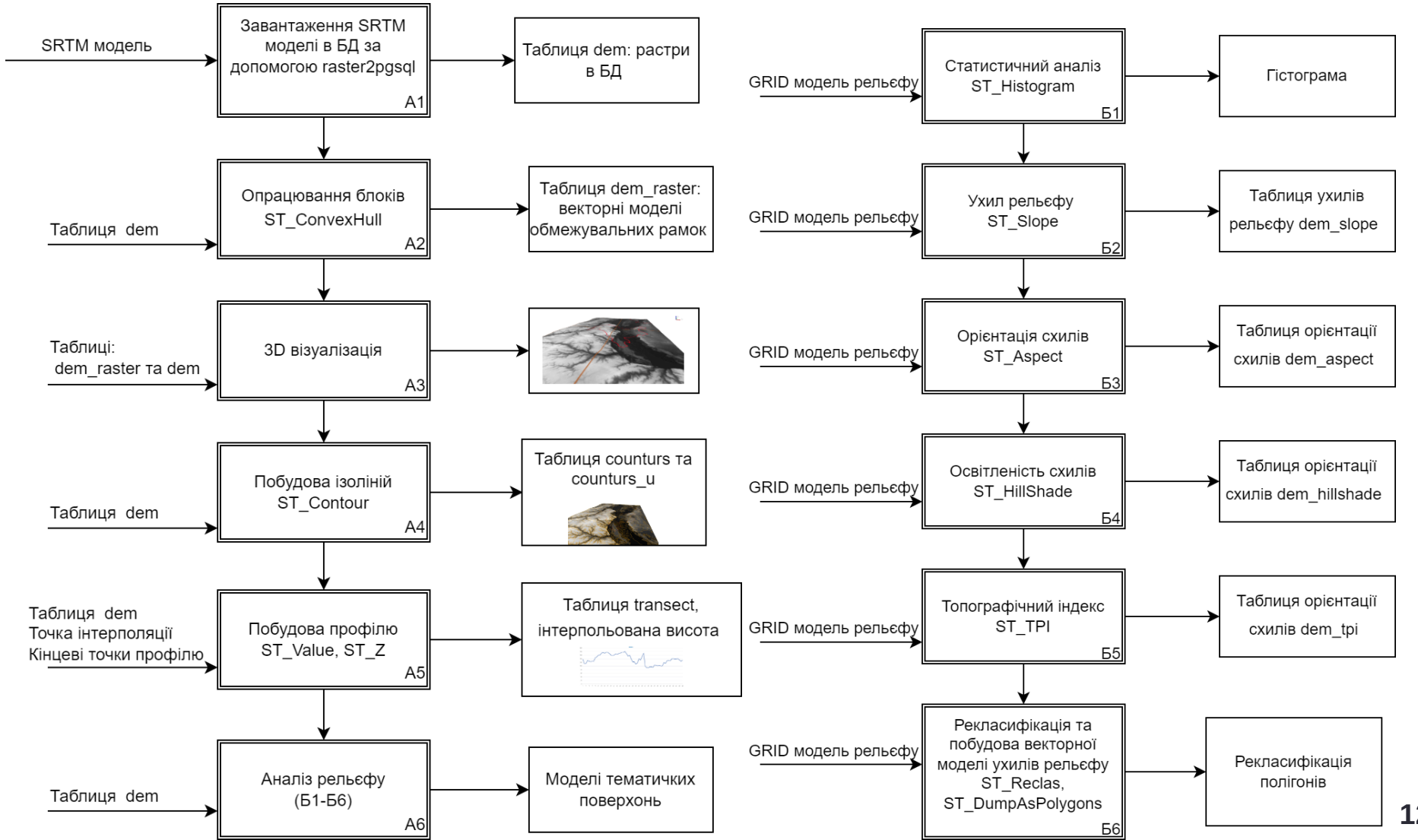
Створення TIN-моделі:



Структурно-функціональна схема ГІС моделювання рельєфу з використанням БГД

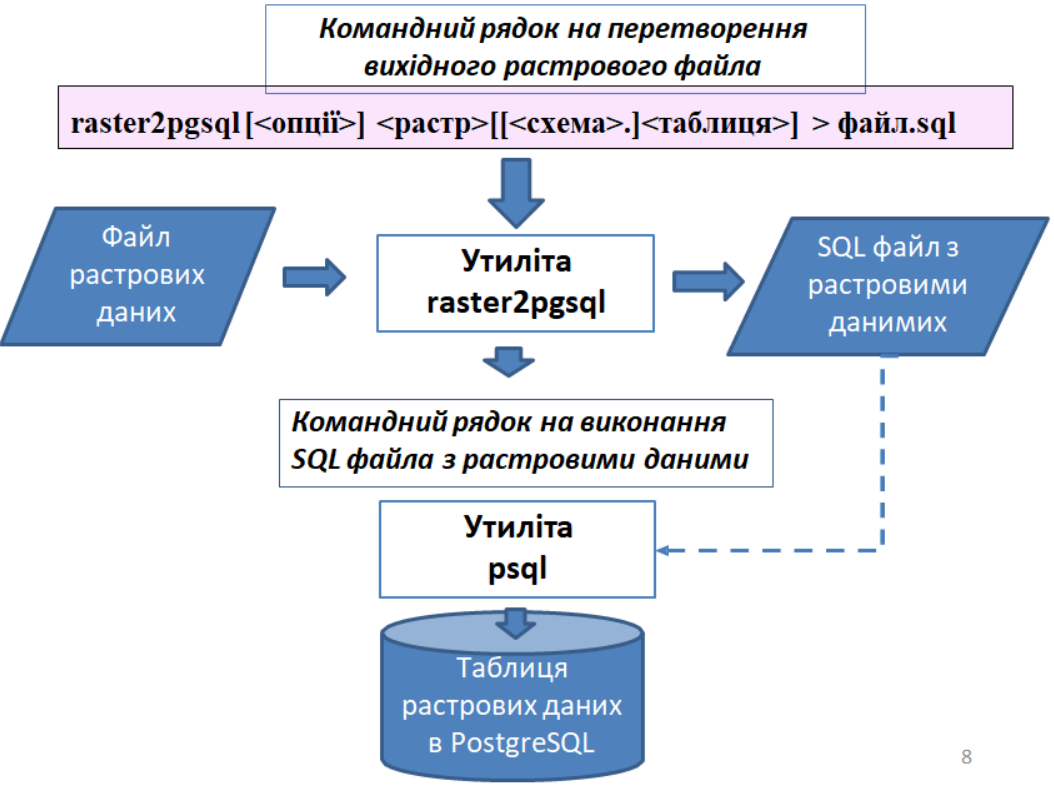


Технологічна схема дослідження використання GRID-моделі в PostGIS



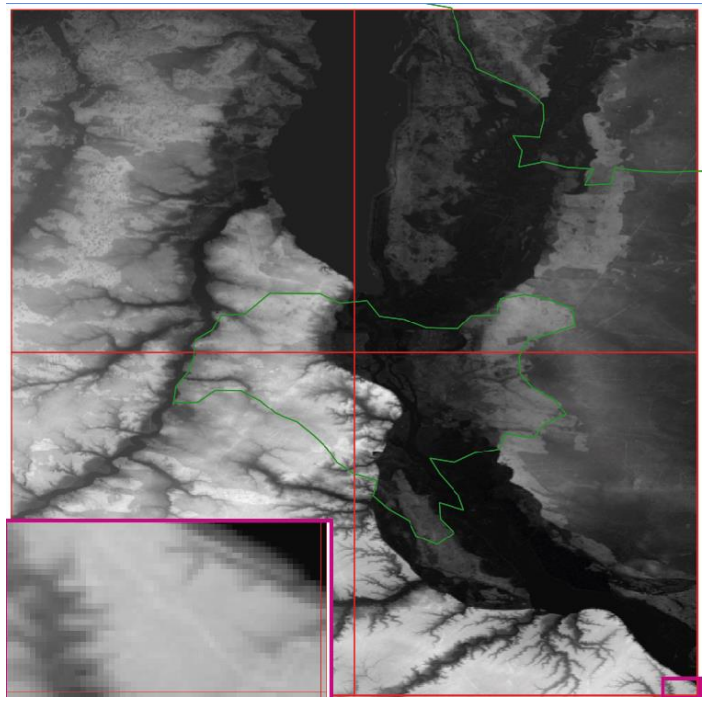
Завантаження GRID-моделі в PostgreSQL/PostGIS та запит на створення рамок блоків растру

Схема завантаження растру:



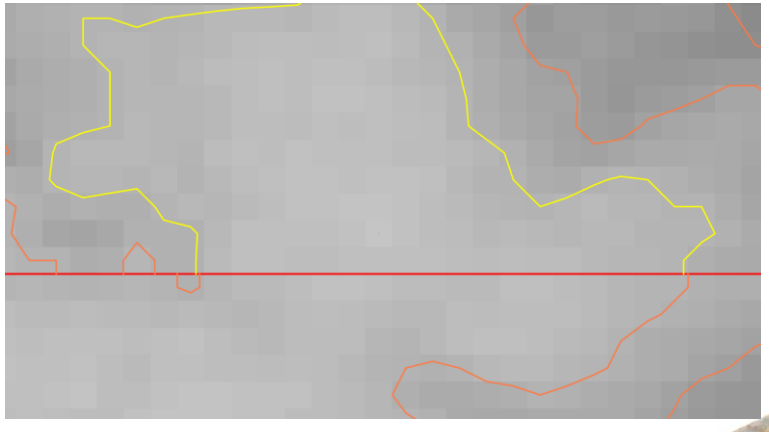
Запит на створення рамок блоків растру:

```
CREATE OR REPLACE VIEW
dem_rasters AS SELECT rid,
ST_ConvexHull(rast) AS geom
FROM dem;
```

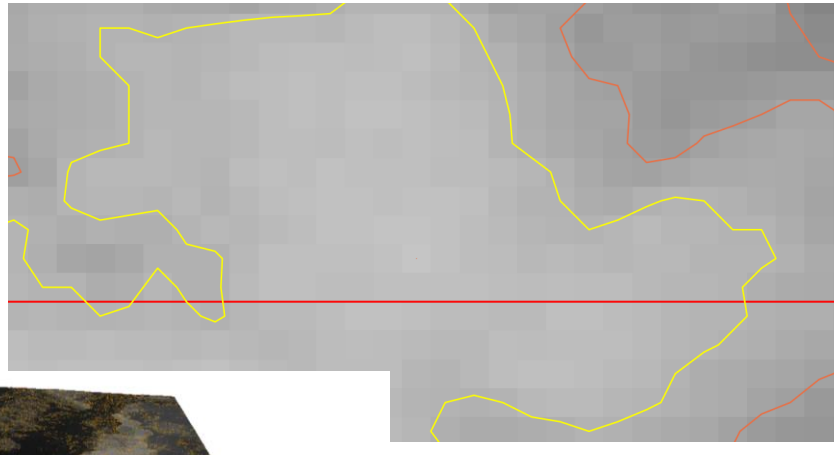


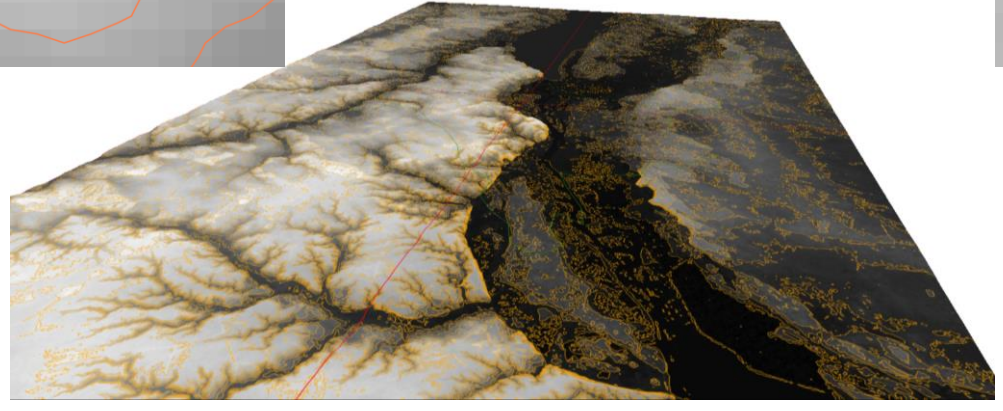
Запиту на побудову ізоліній та 3D візуалізація растру

```
CREATE TABLE contours AS
SELECT (ST_Contour(
 rast,
 bandnumber => 1,
 level_interval => 10)).*
 FROM dem;
```



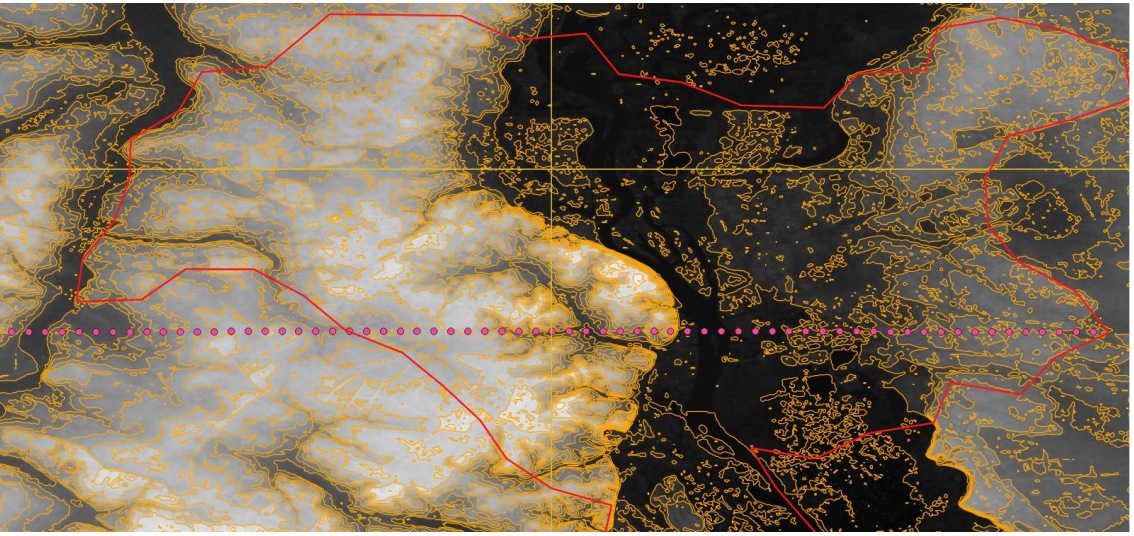
```
CREATE TABLE contours_u AS
SELECT (ST_Contour(
ST_Union(rast),
bandnumber => 1,
level_interval => 10)).*
FROM dem;
```

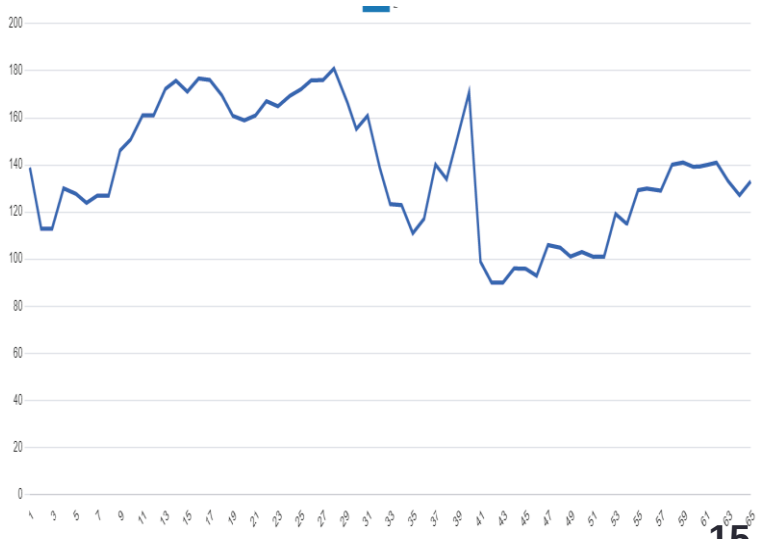




Створення профілю рельєфу з дослідженням методів інтерполяції

```
CREATE TABLE transect AS
WITH transect AS (
SELECT ST_Segmentize(ST_GeogFromText('LINESTRING(30.20 50.41, 30.80 50.41)'), 1000)::geometry AS geom ), rast AS (
SELECT ST_Union(dem.rast) AS rast FROM dem JOIN transect ON ST_Intersects(transect.geom, ST_ConvexHull(dem.rast)) ), z AS (
SELECT (ST_DumpPoints(ST_SetZ(rast.rast, transect.geom, resample => 'bilinear'))).* FROM rast CROSS JOIN transect )
SELECT round(ST_Z(geom)) AS z, geom FROM z;
```





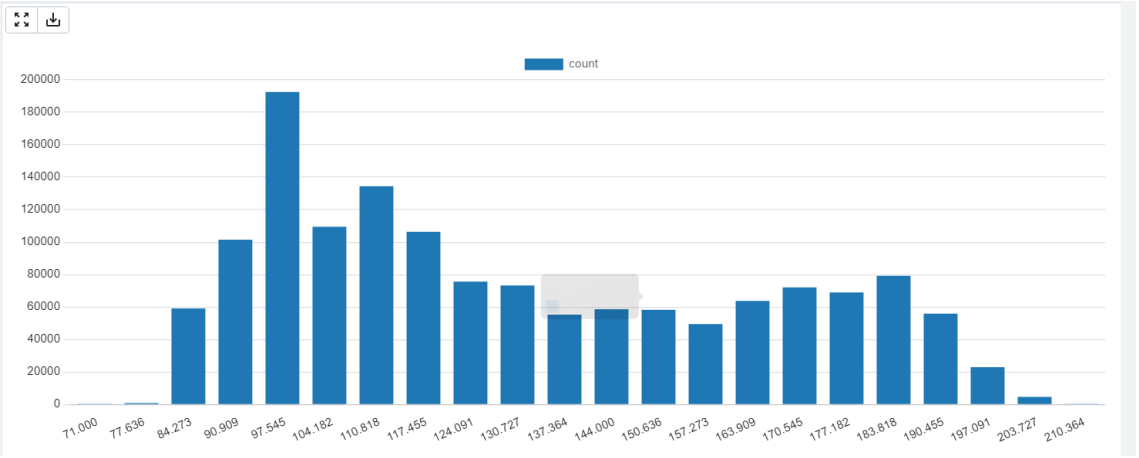
Узагальнення виконання запитів опрацювання GRID-моделі в PostgreSQL/PostGIS

Назва таблиці	Назва основної функції	Назва додаткових функцій	Число створених елементів моделі	Час виконання
dem_rasters	ST_ConvexHull	відсутня	9	53 мсек
counturs	ST_Contour	відсутня	17401	896 мсек
counturs_u	ST_Contour	ST_Union	16782	1 сек 238 мсек
pt (найближчий сусід)	ST_Value	ST_GeomFromText, ST_Intersects, ST_ConvexHull	1	52 мсек
pt (білінійна інтерполяція)	ST_Value	ST_GeomFromText, ST_Intersects ST_ConvexHull	1	93 мсек
transect	ST_Z	ST_Segmentize, ST_GeogFromText, ST_Union, ST_Intersects, ST_ConvexHull, ST_DumpPoints	65	284 мсек

Висновок: Наведені узагальнені дані засвідчують, що функції, які реалізовані для опрацювання GRID-моделі в PostGIS є ефективними. Найскладніший запит виконався трохи більше ніж за одну секунду, при цьому растр містив 1 442 401 пікселів, що свідчить про високий рівень продуктивності та оптимізацію використаних функцій.

Типовий запит на побудови гістограми висот рельєфу за GRID

```
WITH hist AS ( SELECT
(ST_Histogram(ST_Union(rast),1)) .*
from dem
WHERE filename='N50E030.hgt')
SELECT
round(min::numeric,3) AS min,
round(max::numeric,3) AS max, count,
round(percent::numeric,2) AS percent
FROM hist
ORDER BY min;
```

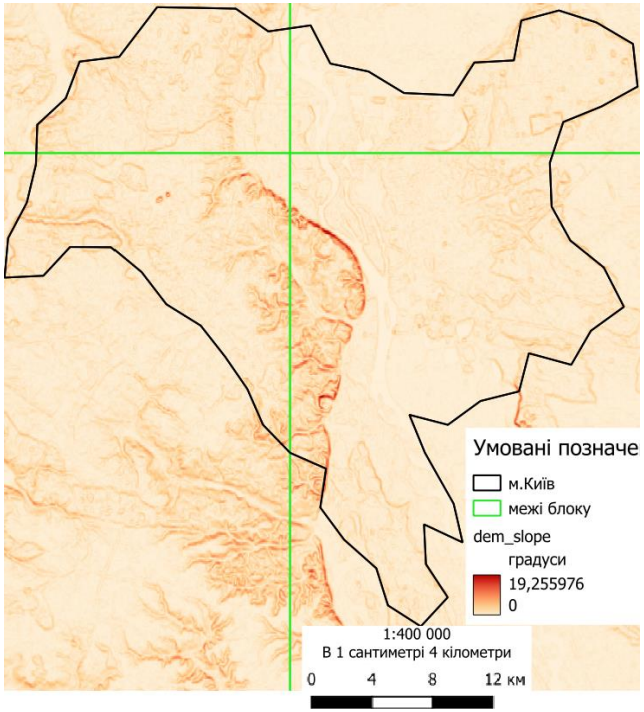


	min numeric	max numeric	count bigint	percent numeric
1	71.000	77.636	1	0.00
2	77.636	84.273	818	0.00
3	84.273	90.909	59153	0.04
4	90.909	97.545	101544	0.07
5	97.545	104.182	192505	0.13
6	104.182	110.818	109415	0.08
7	110.818	117.455	134449	0.09
8	117.455	124.091	106345	0.07
9	124.091	130.727	75645	0.05
10	130.727	137.364	73242	0.05
11	137.364	144.000	55364	0.04
12	144.000	150.636	58605	0.04
13	150.636	157.273	58357	0.04
14	157.273	163.909	49453	0.03
15	163.909	170.545	63748	0.04
16	170.545	177.182	72066	0.05
17	177.182	183.818	68950	0.05
18	183.818	190.455	79213	0.05
19	190.455	197.091	55862	0.04
20	197.091	203.727	22924	0.02
21	203.727	210.364	4641	0.00
22	210.364	217.000	101	0.00

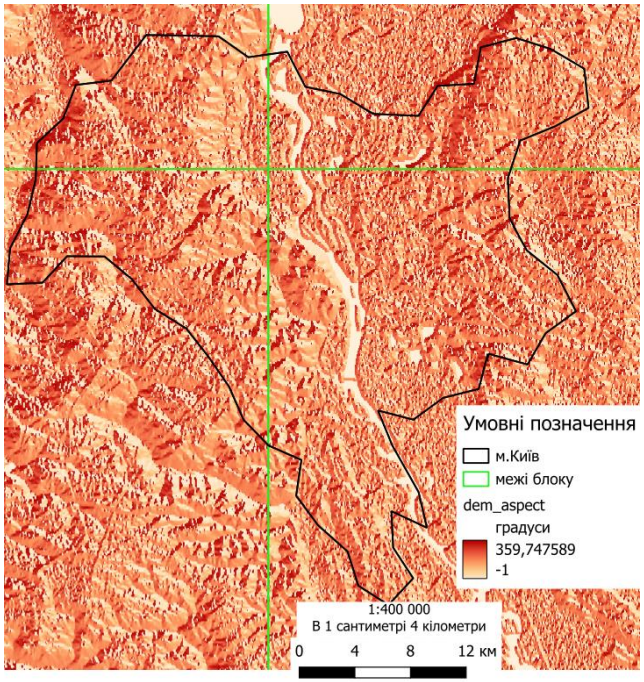
Total rows: 22 of 22 Query complete 00:00:00.885

Типовий запит на створення поверхні ухилів та орієнтації схилів

```
CREATE TABLE dem_slope AS
SELECT rid, ST_Slope(rast::raster,
'1'::int,'32BF'::text,'DEGREES'::text,'
111120'::double precision) AS rast
FROM dem
WHERE filename='N50E030.hgt'
```



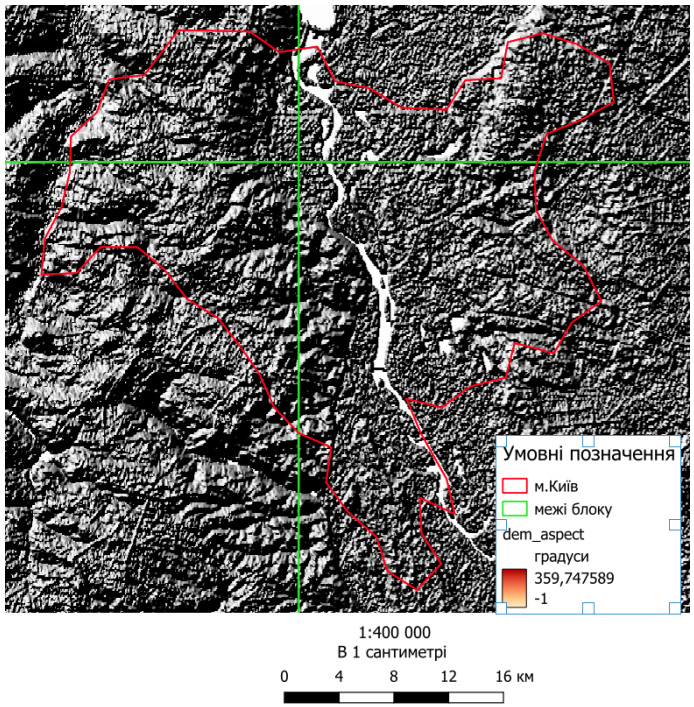

```
CREATE TABLE dem_aspect AS
SELECT rid,
ST_Aspect(rast::raster,'1'::int,'32BF'
::text,'DEGREES'::text,'1'::boolean)
AS rast  FROM dem
WHERE filename='N50E030.hgt'
```

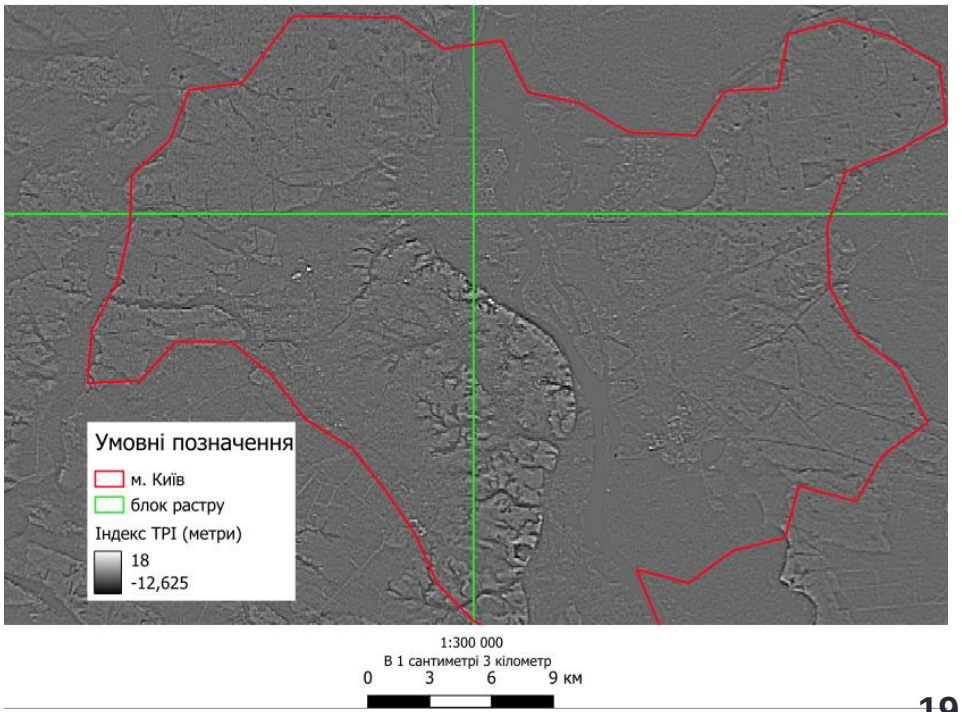


Типові запити для побудови освітленості схилів та індексу TPI

```
CREATE TABLE dem_hillshade AS
SELECT rid ST_HillShade(rast::raster, 1, '32BF'::text, 315, 45, 255, 1.0, FALSE), AS rast
FROM dem
WHERE filename='N50E030.hgt'
```

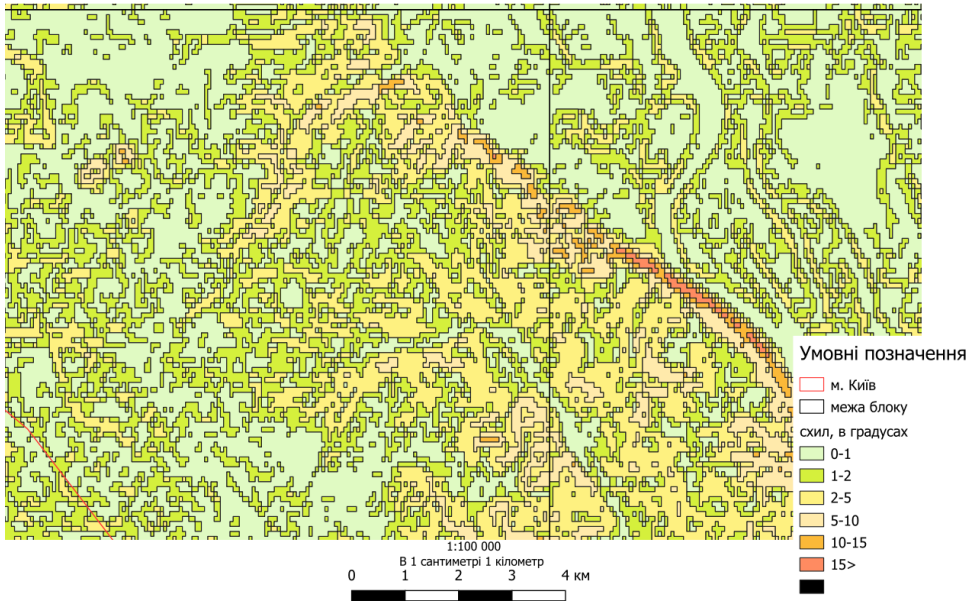
```
CREATE TABLE dem_tpi AS
SELECT rid, ST_TPI (a.rast, 1) AS rast
FROM dem AS a
WHERE filename='N50E030.hgt'
```

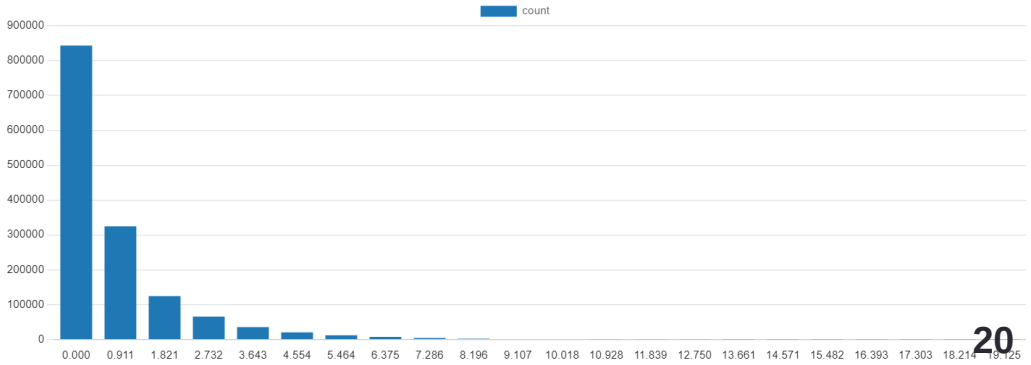




Автоматична векторизація рекласифікованої GRID моделі ухилу поверхні рельєфу

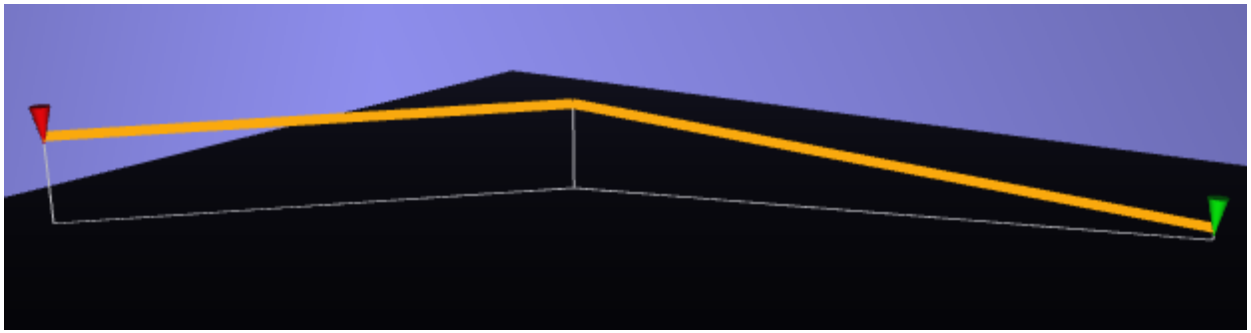
```
CREATE TABLE dem_reclassmpoly AS
SELECT val, geom AS geom
FROM (
  SELECT poly.*
  FROM dem, LATERAL
  ST_DumpAsPolygons(
    ST_Reclass(
      (ST_SLOPE(rast::raster, '1'::int,
'32BF'::text, 'DEGREES'::text,
'111120'::double precision))::raster,
      '1'::int, '0-1):1, 1-2):2, 2-5):3, 5-10):4,
10-15):5, [15-90):6'::text, '32BF'::text
    )
  ) AS poly
  WHERE dem.filename =
'N50E030.hgt') AS foo;
```



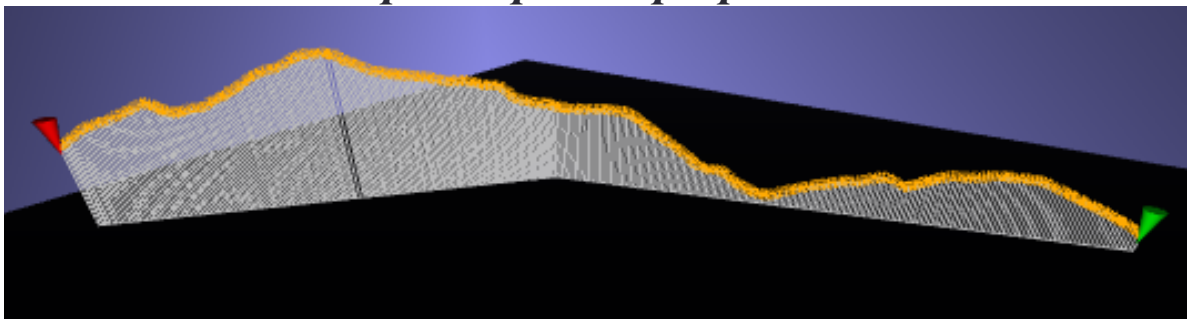


Типи моделей побудови 3D ліній на GRID-моделі

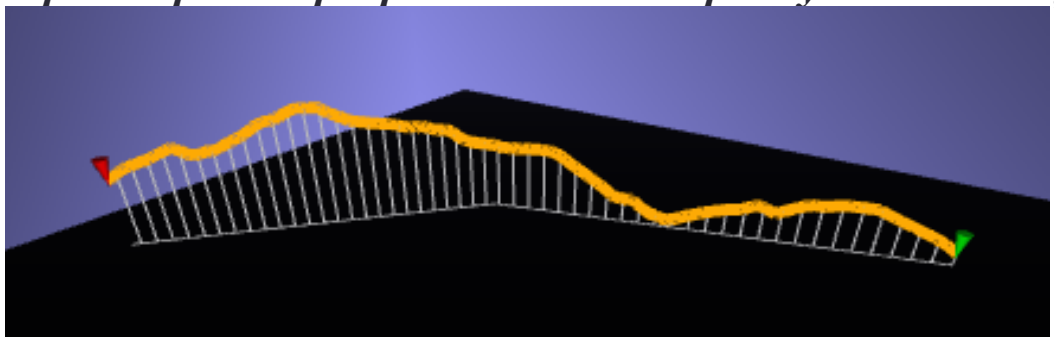
Моделювання з просторовим розрізненням вхідної 2D геометрії:



Моделювання з просторовим розрізненням GRID:

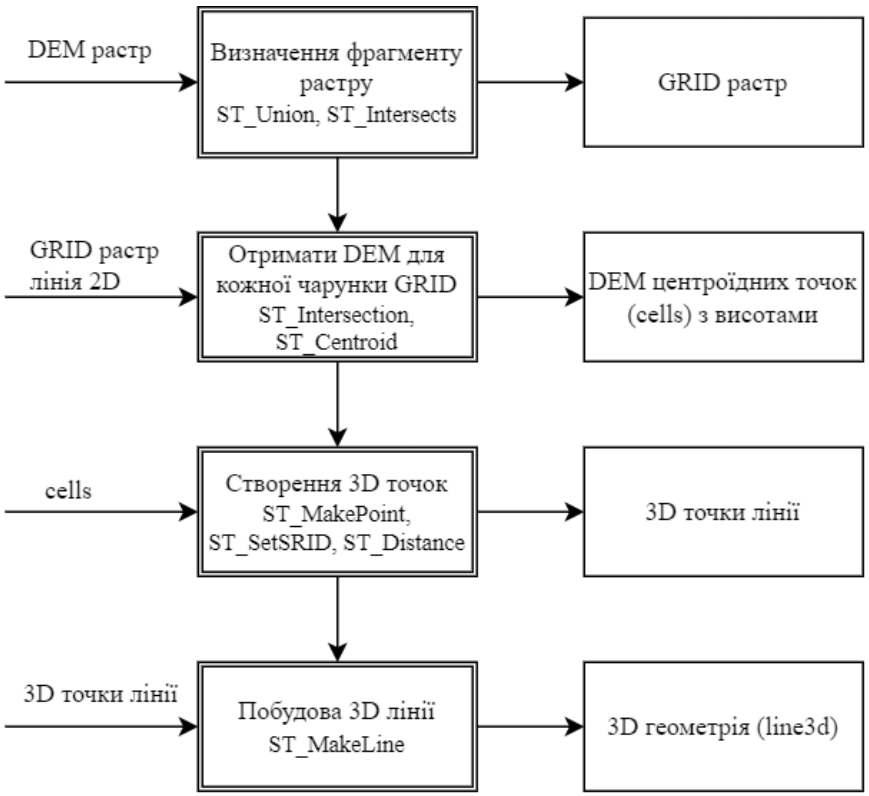


Моделювання з просторовим розрізненням інтервалу сегментації вхідної 2D лінії:



Функція побудови 3D ліній з розрізненням GRID-моделі

Схема алгоритму функції



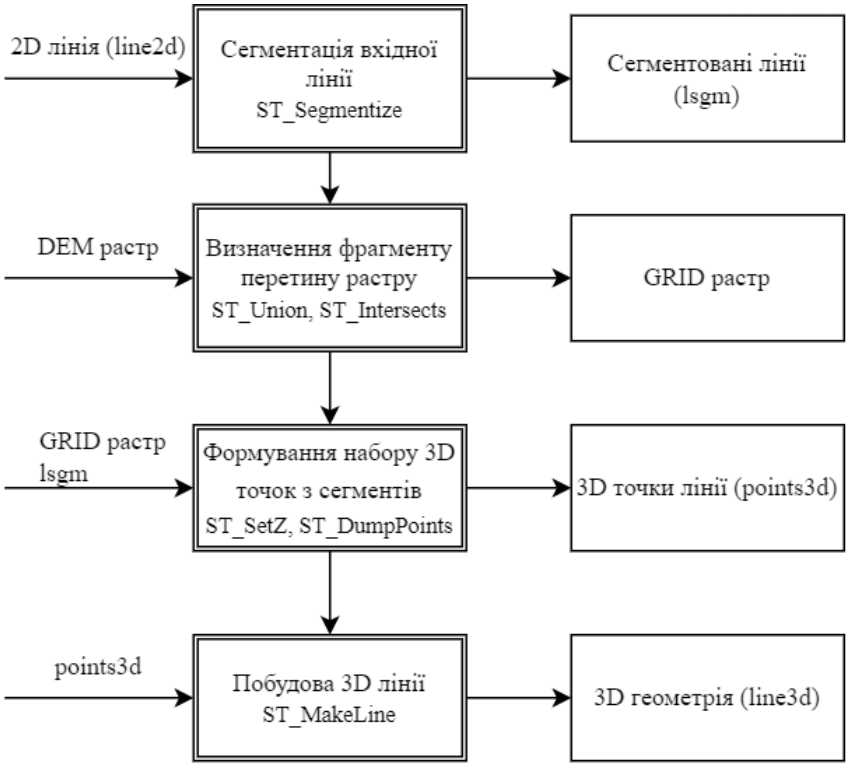
Текст SQL-функції:

```
CREATE OR REPLACE FUNCTION _line3d(line2d geometry)
RETURNS geometry AS
$BODY$
DECLARE
line3d geometry;
BEGIN
WITH grid AS
-- визначення фрагменту растру, перетинається з вхідною лінією
(SELECT ST_Union(dem.rast) AS rast  FROM dem
WHERE ST_Intersects(line2d, ST_ConvexHull(dem.rast))),
cells AS
-- отримати DEM для кожної чарунки GRID, що перетинається з лінією
(SELECT ST_Centroid((ST_Intersection(grid.rast, line2d)).geom) AS geom,
(ST_Intersection(grid.rast, line2d)).val AS val
FROM grid
WHERE ST_Intersects(grid.rast, line2d)),
-- створення 3D точок для перетину лінії з чарунками GRID
points3d AS (SELECT ST_SetSRID(ST_MakePoint(ST_X(cells.geom),
ST_Y(cells.geom), val), 4326) AS geom FROM cells
ORDER BY ST_Distance(ST_StartPoint(line2d), cells.geom))
-- побудова 3D лінії для 3D точок та повернення результату
SELECT ST_MakeLine(geom) INTO line3d FROM points3d;
 RETURN line3d;END;
                $BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Висновок: Функція забезпечує високу точність, але час обчислення є надто великим. Такий значний час обчислень пов'язаний із процедурою пошуку точок перетину вихідної лінії з прямокутниками пікселів GRID-моделі рельєфу. Прямокутники створюються для всього растру без просторового індексування, що потребує перебирати сотні тисяч пікселів, навіть для короткої лінії.

Функція побудови 3D ліній з розрізненням інтервалу сегментації вхідної 2D лінії

Схема алгоритму функції



Висновок: Функція ***_line3ds(line2d geometry, sgm_length float)*** виконується за 230–670 мс на лінію, що значно швидше порівняно з ***_line3d(line2d geometry)*** (~1 хв. 22 сек), завдяки прямій інтерполяції координат z без аналізу перетинів з прямокутниками пікселів GRID

Текст SQL-функції:

```
CREATE OR REPLACE FUNCTION _line3ds(line2d geometry,sgm_length
float)
  RETURNS geometry AS
$BODY$
DECLARE
line3d geometry;
          BEGIN
-- сегментація вхідної лінії з максимальною заданою довжиною сегмента
WITH lsgm AS (
 SELECT ST_Segmentize(line2d::geography,sgm_length)::geometry AS
geom),
-- визначення фрагменту dem, що перетинається з лінією
grid AS (
 SELECT ST_Union(dem.rast) AS rast
  FROM dem
  JOIN lsgm
  ON ST_Intersects(lsgm.geom, ST_ConvexHull(dem.rast))),
-- формування набору 3D точок із точок сегментів лінії та Z растру
points3d AS (
  SELECT (ST_DumpPoints(ST_SetZ(grid.rast, lsgm.geom, resample =>
'bilinear'))).*
 FROM grid
  CROSS JOIN lsgm)
-- формування 3d лінії
SELECT ST_MakeLine(points3d.geom) INTO line3d FROM points3d;
    RETURN line3d;
END;
          $BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

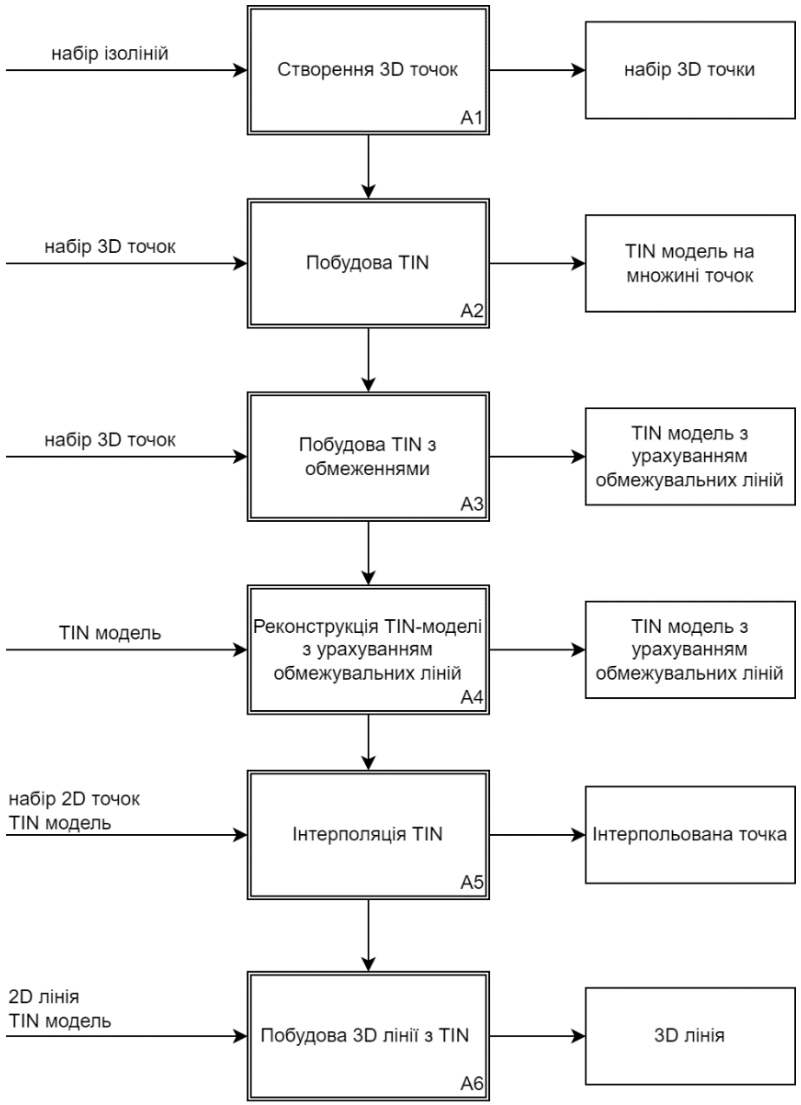
Функція обчислення довжини 3D лінії з просторовим розрізненням інтервалу сегментації вхідної 2D лінії

```
CREATE OR REPLACE FUNCTION _line3dslg(line2d geometry,sgm_length
float)
  RETURNS double precision AS
$BODY$
DECLARE
line3d geometry;
lg3d double precision;
          BEGIN
-- сегментація вхідної лінії з максимальною заданою довжиною сегмента
WITH lsgm AS (
 SELECT ST_Segmentize(line2d::geography,sgm_length)::geometry AS geom),
-- визначення фрагменту dem, що перетинається з лінією
grid AS (
 SELECT ST_Union(dem.rast) AS rast
  FROM dem
  JOIN lsgm
  ON ST_Intersects(lsgm.geom, ST_ConvexHull(dem.rast))),
-- формування набору 3D точок із точок сегметів лінії та Z растру
points3d AS (
  SELECT (ST_DumpPoints(ST_SetZ(grid.rast, lsgm.geom, resample =>
'bilinear'))).*
 FROM grid
  CROSS JOIN lsgm)
-- формування 3d лінії
SELECT ST_MakeLine(points3d.geom) INTO line3d FROM points3d;
-- обчислення довжини на еліпсоїді
 lg3d=ST_LengthSpheroid(line3d,
'SPHEROID["WGS 84",6378137,298.257223563]');
   RETURN lg3d;
END;
            $BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Висновок: Результати обчислень довжин ділянок вулиць для міста Києва показали, що різниця між довжинами 2D і 3D ліній, визначених на еліпсоїді, є незначною, до декількох метрів. Водночас, довжини на еліпсоїді та в проекції Меркатора суттєво відрізняються — до сотень і тисяч метрів.

Назва вулиці	Довжина вулиць, м			Різниця довжини	
	2D, на еліпсоїді	3D, на еліпсоїді	2D, на проєкції	на еліпсоїді	порівняно з L2D на проєкції
"Круглоуніверситетська"	609.278	611.853	954.811	2.575	342.958
"Велика Васильківська"	3704.338	3706.724	5803.161	2.386	2096.437
"Тараса Шевченка бульвар"	1877.911	1880.295	2942.979	2.384	1062.684

Технологічна схема дослідження використання TIN-моделі в PostGIS



A1) отримання 3D точок на основі набору ізоліній;

A2) побудова TIN-моделі без обмежень для набору 3D точок з використанням функції PostGIS ST_DelaunayTriangles;

A3) побудова TIN-моделі з врахуванням обмежувальних ліній з використанням функції PostGIS ST_ConstrainedDelaunayTriangles;

A4) розробка і використання функції для реконструкції TIN-моделі Делоне за набором обмежувальних ліній;

A5) реалізація функції інтерполяції висоти довільної 2D точки з використанням TIN-моделі

A6) використання TIN-моделі для підняття 2D лінії до 3D, враховуючи рельєф.

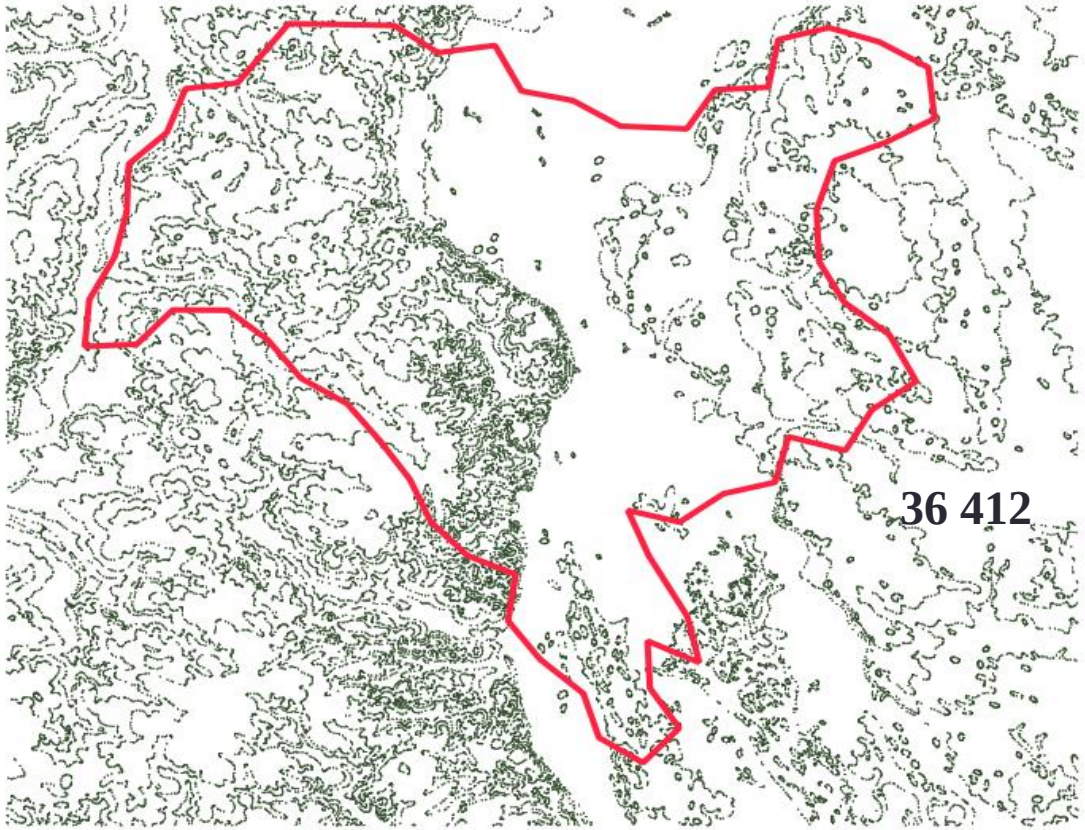
Формування набору 3D точок

Створення таблиці 3D точок:

```
CREATE TABLE pts3d(
    gid serial PRIMARY KEY,
    geom geometry(POINTZ,4326) );
```

Заповнення таблиці pts3d на основі 2D точок з атрибутом висоти:

```
INSERT INTO pts3d (gid, geom)
SELECT gid, ST_Force3DZ(p.geom, p.hg)
FROM point_kyiv AS p;
```



Запит тестування:

```
SELECT st_astext(geom) FROM pts3d LIMIT 4;
```

Результат:_____

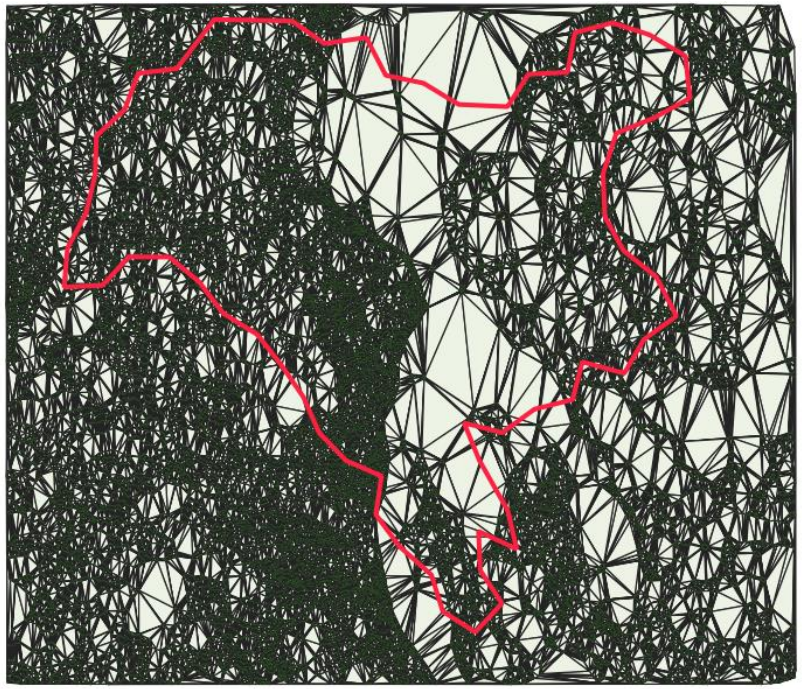
"POINT Z (30.742868999999992 50.595043000000004 120)"
"POINT Z (30.742466 50.594674999999995 120)"
"POINT Z (30.741374000000004 50.594024000000005 120)"
"POINT Z (30.74094 50.593279 120)"

Запити формування TIN-моделі та вивід окремих трикутників

Запит на побудову тріангуляції Делоне:

```
CREATE TABLE tin_dln AS
WITH pts AS
(
SELECT geom AS pt
FROM pts3d
)
SELECT ST_DelaunayTriangles(ST_Union(pt::geometry), 0.0, 0) AS
the_geom
FROM pts;
```

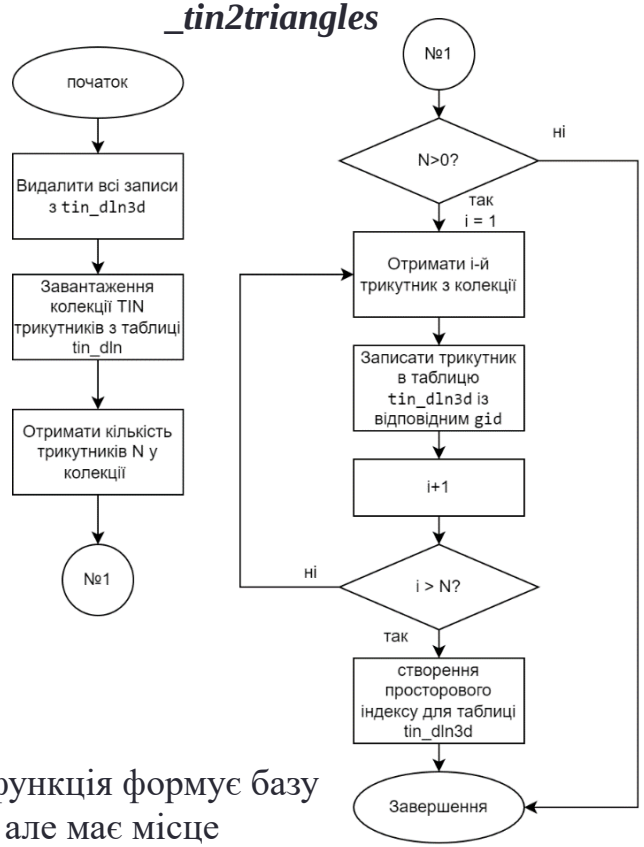
Висновок запиту: створено 71 940 трикутників за 808 msec, але вся TIN знаходиться в єдиному записі.



Розроблення таблиці вивідуй трикутників:

```
CREATE TABLE tin_dln3d(
   gid serial PRIMARY KEY,
   geom geometry(POLYGONZ,4326) );
```

Схема алгоритму функції *_tin2triangles*



Оцінка алгоритму: функція формує базу окремих трикутників але має місце великих затрат часу

Типові запити для формування TIN моделей з обмеженнями

запит типу PLT:

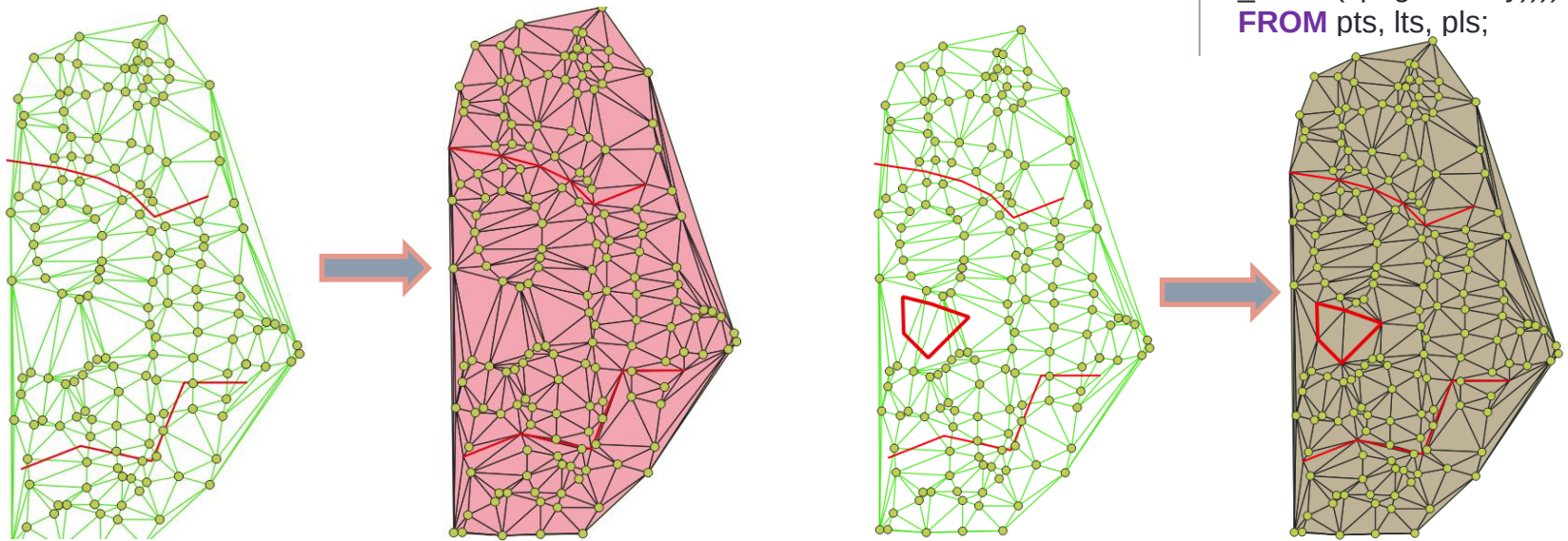
```
CREATE TABLE tin_dlc AS
WITH pts AS
(SELECT geom AS pt FROM points),
lts AS
(SELECT geom AS cl FROM constr_line)
SELECT
ST_ConstrainedDelaunayTriangles(ST_Colle
ct(ST_Union(pt::geometry),
ST_Union(cl::geometry))) AS geom
FROM pts, lts;
```

запит типу PPT :

```
CREATE TABLE tin_dlc AS
WITH pts AS
(SELECT geom AS pts FROM points),
pls AS
(SELECT geom AS cp FROM
constr_polygon)
SELECT
ST_ConstrainedDelaunayTriangles(ST_
Collect(ST_Union(pt::geometry),
ST_Collect(ST_Union(cp::geometry)))
AS geom
FROM pts, pls;
```

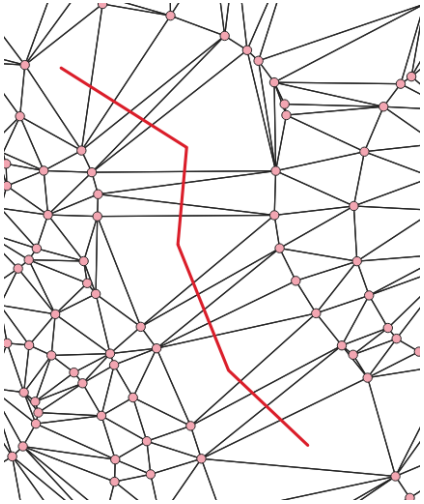
запит типу PLPT:

```
CREATE TABLE tin_dlc AS
WITH pts AS
(SELECT geom AS pts FROM points),
lts AS
(SELECT geom AS cl FROM
constr_line),
pls AS
(SELECT geom AS cp FROM
constr_polygon)
SELECT
ST_ConstrainedDelaunayTriangles(ST
_Collect(ST_Union(pt::geometry),
ST_Collect(ST_Union(cl::geometry),ST
_Union(cp::geometry)))) AS geom
FROM pts, lts, pls;
```

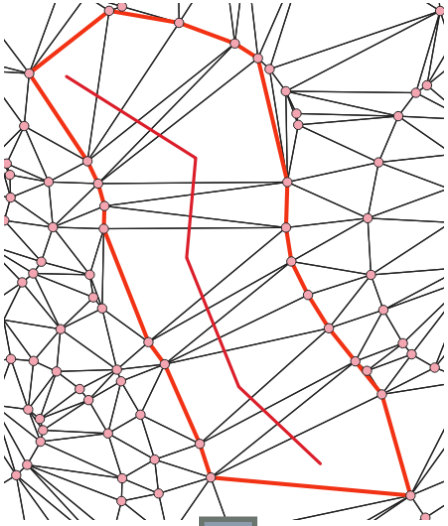


Реалізація функції реконструкції TIN-моделі за набором структурних ліній

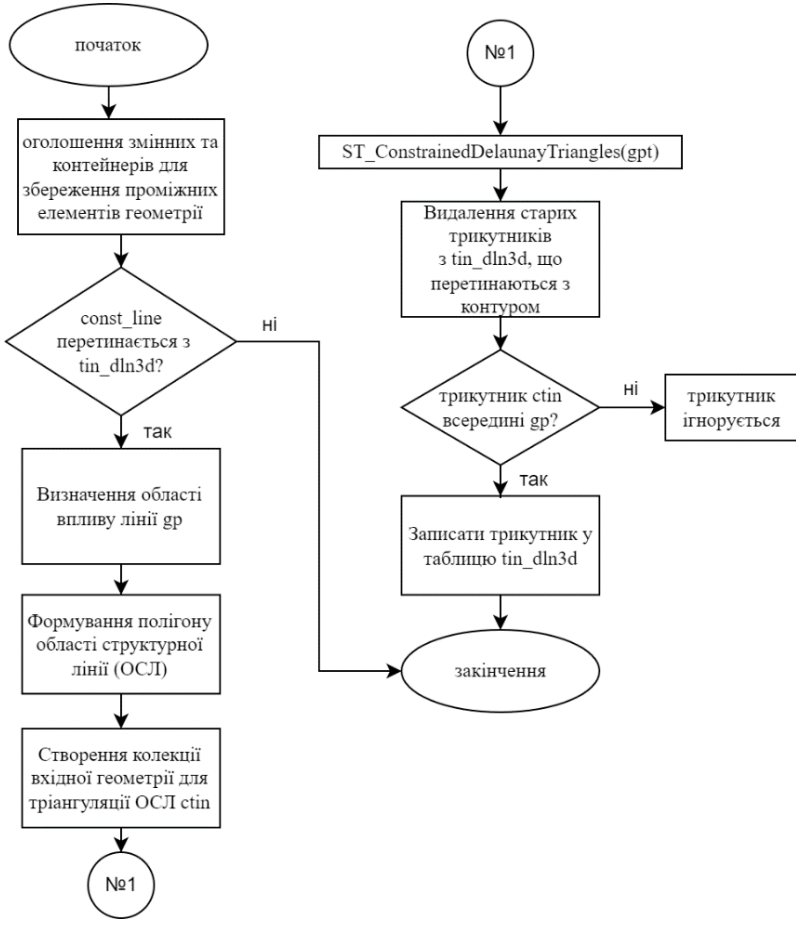
Структурна лінія:

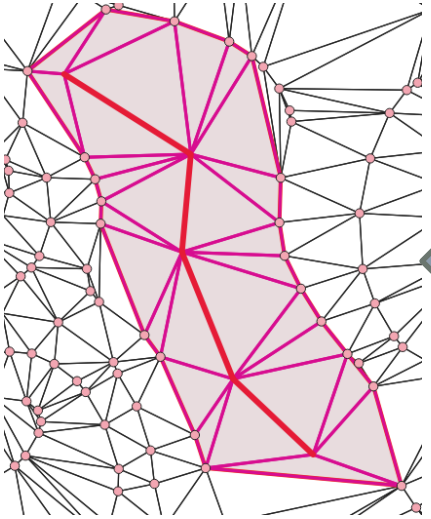


Визначення полігону області реконструкції та її вилучення :

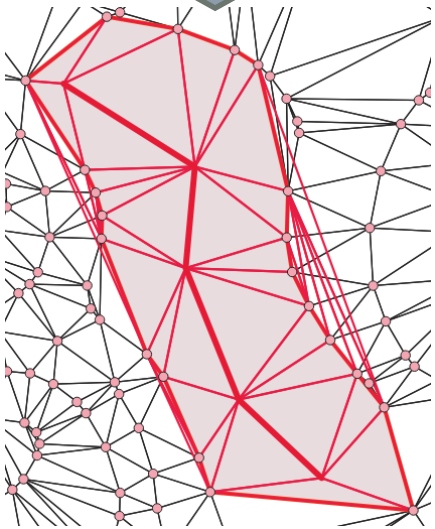


Узагальнена схема алгоритму функції *_rectin_cl()*





Обрізання зайвих трикутників та вставлення в TIN:



Тріангуляція області реконструкції:

Функція формування набору трикутників 3DTIN моделі рельєфу та її реконструкції

Функції _tin2triangles перетворює колекцію трикутників типу TIN в окремі трикутники

```
CREATE OR REPLACE FUNCTION _tin2triangles()
  RETURNS text AS
$BODY$
DECLARE
k record;
coltin geometry;
BEGIN
delete from tin_dln3d;
coltin = (SELECT geom FROM tin_dln);
FOR k in 1..ST_NumGeometries(coltin)
LOOP
insert into tin_dln3d(gid,geom)
            values (k,ST_GeometryN(coltin,k));
END LOOP;
CREATE INDEX tin_dln3d_geom_idx ON
            tin_dln3d USING GIST (geom);
return 'Done';
END;
             $BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Функція _rectin_cl() призначена для реконструкції TIN за обмеженнями

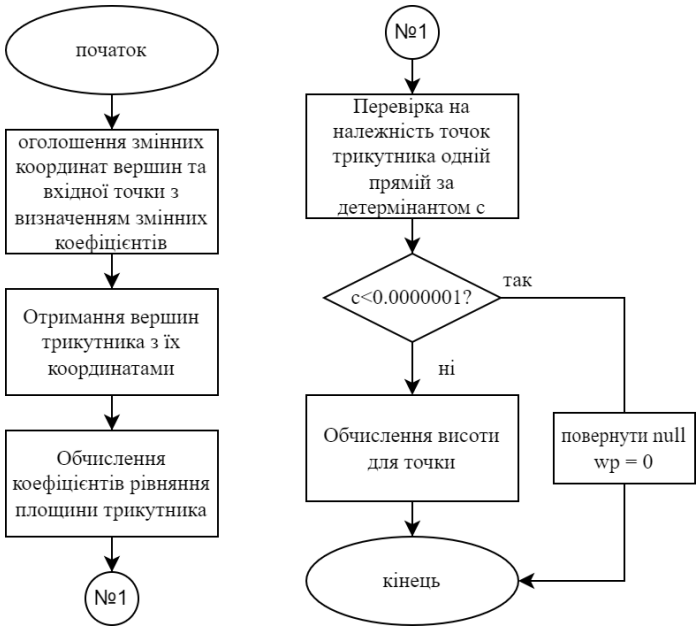
```
CREATE OR REPLACE FUNCTION _rectin_cl()
  RETURNS text AS
$BODY$
DECLARE
j record;
k record;
gpt geometry;
ctin geometry;
ontr geometry;
BEGIN
FOR j IN (SELECT gid,geom FROM constr_line)
  LOOP
gp=(SELECT ST_Union(geom) FROM tin_dln3d AS a
      WHERE ST_Intersects(j.geom, a.geom));
gpt=ST_Collect(j.geom,gp);
ctin=ST_ConstrainedDelaunayTriangles(gpt);
DELETE FROM tin_dln3d AS a
WHERE ST_Intersects(j.geom, a.geom);
FOR k in 1..ST_NumGeometries(ctin)
LOOP
ontr=ST_GeometryN(ctin,k);
IF ST_Within(ontr,gp) THEN
  insert into tin_dln3d (geom) values (ST_GeometryN(ctin,k);
END IF;
END LOOP;
END LOOP;
return 'Done';
           END;
           $BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Реалізація функції інтерполяції висоти довільної 2D точки за 3DTIN моделлю рельєфу

Функція _point_tin_value (triangle3d, point2d) повертає значення висоти для точки point2d за трикутником triangle3d поверхні TIN

```
CREATE OR REPLACE FUNCTION _point_tin_value
(triangle3d geometry, point2d geometry)
RETURNS double precision AS
$BODY$
DECLARE
x1 double precision;
y1 double precision;
w1 double precision;
x2 double precision;
y2 double precision;
w2 double precision;
x3 double precision;
y3 double precision;
w3 double precision;
xp double precision;
yp double precision;
tolerance double precision = 0.0000001;
a double precision;
b double precision;
c double precision;
pts geometry; -- контейнер мультиточки для вершин трикутника
p1 geometry; -- контейнери для точок вершин трикутника
p2 geometry;
p3 geometry;
BEGIN
-- отримання (витягування) точок та їх координат із вхідної геометрії
pts = ST_Points(triangle3d);
p1 = ST_GeometryN(pts, 1);
p2 = ST_GeometryN(pts, 2);
p3 = ST_GeometryN(pts, 3);
x1 = ST_X(p1);
y1 = ST_Y(p1);
w1 = ST_Z(p1);
x2 = ST_X(p2);
y2 = ST_Y(p2);
w2 = ST_Z(p2);
x3 = ST_X(p3);
y3 = ST_Y(p3);
w3 = ST_Z(p3);
xp = ST_X(point2d);
yp = ST_Y(point2d);
-- обчислення детермінантів для рівняння площини трикутника
a := ((y2-y1) * (w3-w1) - (w2-w1) * (y3-y1));
b := ((w2-w1) * (x3-x1) - (x2-x1) * (w3-w1));
c := ((x2-x1) * (y3-y1) - (y2-y1) * (x3-x1));
-- перевірка щодо належності точок трикутника одній прямій
if abs(c) < tolerance THEN return null;
end if;
-- обчислення та повернення висоти для вхідної точки
return (w1 - (((a * (xp-x1))+(b * (yp-y1)))/c));
END;
$BODY$
LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Загальна схема алгоритму функції **_point_tin_value (triangle3d, point2d)**



Запит на тестування функції:
_point_tin_value(triangle3d, point2d):

```
SELECT _point_tin_value(
'TIN Z(
((1000.0 1000.0 100.0,2000.0 1800.0 500.0,
2000 1000.0 100.0,1000.0 1000.0
100.0)))'::geometry,
'POINT(1250 1100)'::geometry);
```

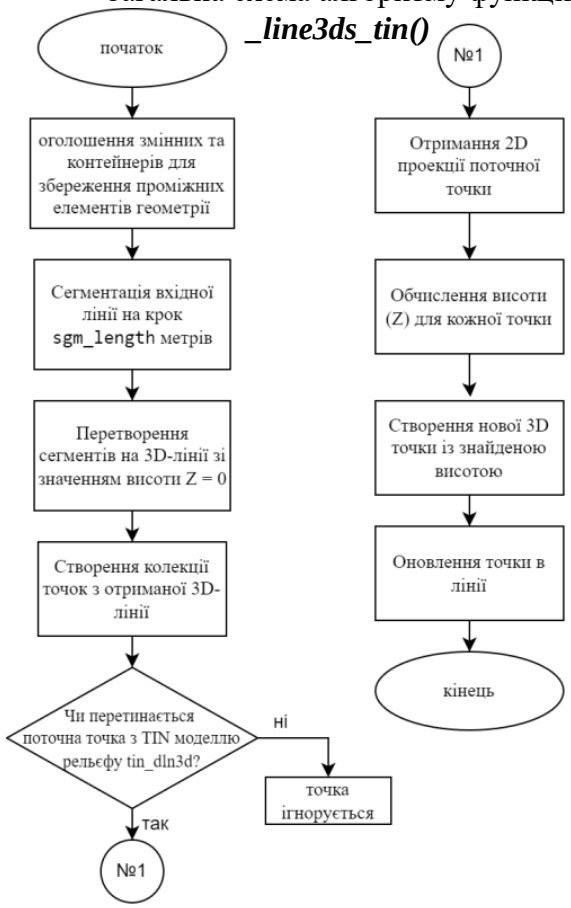
Результат: 150

Реалізація функції побудови 3D ліній з використанням 3DTIN моделлю рельєфу

Функція: _line3ds_tin(line2d geometry, sgm_length float) повертає 3D геометрію, сформовану для полілінії line2d з точками її сегментації.

```
CREATE OR REPLACE FUNCTION _line3ds_tin(line2d geometry, sgm_length
float)
  RETURNS geometry AS
$BODY$
DECLARE
lsgm geometry;
lsgm3d geometry;
pts geometry;
line3d geometry;
k record;
p2d geometry;
p3d geometry;
zp double precision;
BEGIN
lsgm = ST_Segmentize(line2d::geography,sgm_length)::geometry;
lsgm3d = ST_Force3D(lsgm, 0.0);
pts = ST_Points(lsgm3d); line3d = ST_LineFromMultiPoint(pts);
FOR k in 1..ST_NumGeometries(pts)
LOOP
p2d = ST_Force2D(ST_GeometryN(pts,k));
zp=(SELECT _point_tin_value(tin_dln3d.geom, p2d) FROM tin_dln3d
WHERE ST_Intersects(tin_dln3d.geom,p2d));
p3d = ST_Force3D(p2d,zp);
line3d = ST_SetPoint(line3d,k-1,p3d);
END LOOP;
RETURN line3d;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE STRICT
  COST 100;
```

Загальна схема алгоритму функції ***_line3ds_tin()***



Запит обрахунку бульвару Тараса Шевченка:

```
SELECT ST_AsText (_line3ds_tin(str_test.geom, 50.0))
FROM str_test WHERE str_test.gid = 1;
```

Результат:

Successfully run. Total query runtime: 196 msec

"LINESTRING Z (30.4944945 50.4463704 140,30.4950213 50.4462878 140)"

Порівняння висоти точок і довжин вулиць отриманих за різних моделями рельєфу

Порівняння висоти 3D точок осьових ліній бульвару Тараса Шевченка в Києві за GRID і TIN моделями

№	Координати точок				Відхилення Ztin - Zgrid
	Довгота	Широта	Zgrid, м	Ztin, м	
1	30.5171977	50.4426882	166.184	166.971	0,787
2	30.5172358	50.4426821	166.121	166.956	0,853
3	30.517305	50.442671	166.007	166.928	0,921
4	30.517657027912414	50.44261506621689	165.825	165.966	0,141
5	30.51800905499273	50.442559131371794	166.332	164.835	- 1,497
6	30.51836108124095	50.442503195464745	166.800	163.704	- 2,006
7	30.518713106657042	50.442447258495704	164.208	162.573	- 1,635
8	30.519065131241003	50.442391320046471	161.500	161.443	- 0,057
9	30.519417154992833	50.442335381371755	158.196	160.312	2,116
10	30.5197691779125	50.44227944121686	154.920	160.000	5,080
11	30.5201212	50.4422235	152.851	158.574	5,723

Порівняння загальних довжин осьових ліній вулиць тестового набору

№	Назва вулиці	Довжина вулиці на еліпсоїді, м			L_3Dgrid - L_3D tin
		L_2D	L_3D grid	L_3D tin	
1	Круглоуніверситетська	609.278	611.853	610.987	0.866
2	Велика Васильківська	3704.338	3706.724	3705.428	1.296
3	Тараса Шевченка бульвар	1877.911	1880.295	1879.143	1.153

Статистичний аналіз відхилення висот між GRID і TIN моделями:

- Мінімальне відхилення: 0,057 м
- Середнє абсолютне відхилення: 1,892 м
- Середнє квадратичне відхилення: 2,598 м

Основні причини відхилень:

1. Різні системи відліку висот (GRID: SRTM, TIN: ізолінії рельєфу).
2. Відмінності в просторовому розрізненні моделей.

Висновок:

1.Незважаючи на відмінності значень координат **z** між GRID і TIN моделями рельєфу, **різниця у довжинах 3D ліній** є незначною (від **0,866 до 1,296 м**).

2.Це підтверджує **коректність** розроблених функцій **_point_tin_value** та **_line3ds_tin**.

Загальні висновоки

У роботі на реальних наборах даних проведено обчислювальні експерименти щодо створення цифрових моделей рельєфу в середовищі СКБД PostgreSQL/PostGIS та встановлено що:

а) в PostGIS надаються ефективні засоби для підтримки і аналізу GRID моделей рельєфу на основі спеціального типу даних для растрових моделей та набору функцій побудови поверхонь морфологічних характеристик рельєфу;

б) базові функції PostGIS реалізують тріангуляцію Делоне та тріангуляцію Делене з обмеженнями з наданням результатів у форматі спеціального типу даних TIN як єдиної колекції трикутників, що є раціональним з точки зору реалізації вбудованих прикладних функцій тріангуляції.

2. Для підвищення ефективності використання базових функцій PostGIS моделювання рельєфу в прикладних задачах в роботі реалізовані прикладні SQL функції, які забезпечують:

а) побудову 3D моделей ліній та визначення їх довжини на поверхні з використанням GRID моделі рельєфу;

б) формування набору трикутників TIN моделі рельєфу з використанням базових функцій PostGIS для тріангуляції Делоне та тріангуляції з обмеженнями та побудовою просторового індексу для набору трикутників для оптимізації доступу до них в прикладних задачах моделювання 3D об’єктів та аналізу рельєфу;

в) реконструкцію існуючих TIN моделі рельєфу за набором геопросторових даних структурних ліній рельєфу;

г) визначення висоти довільної 2D точки на поверхні з використанням TIN моделі рельєфу;

д) побудову 3D моделей ліній на поверхні заданої TIN моделлю рельєфу.

3. Коректність запропонованих алгоритмів та реалізованих функцій підтверджено результатами обчислювальних експериментів з побудови TIN моделі рельєфу на дослідну територію міста Києва, моделювання 3D ліній ділянок осьових ліній тестового набору вулиць та визначення їх довжини.