Report

1. UNSW Lower Campus

1.1 Raw (Original) Voxel Datasets

Raw Data	Size	#Voxels	Description
dtmbot.xyz	9.22 GB	641,624,355	A terrain with holes, in which the buildings fit
tree.xyz	906 MB	59,640,000	Tree in lower campus
bld1-54.xyz (except for 26 and 46)	3.52GB	241,613,693 in total and 4,646,418 per building	52 buildings in lower campus
be.xyz	249 MB	17,460,029	Built Environment (H13)
blockhouse.xyz	45.4 MB	3,392,202	Blockhouse (G6)
dalton.xyz	25.5 MB	1,887,512	Dalton (F12)
quadrangle.xyz	43.9MB	3,161,733	Quadrangle (E15)
roundhouse.xyz	79.9MB	6,037,174	Roundhouse (E6)
scithe.xyz	17.2MB	1,231,821	Science Theatre (F13)

Note that:

- For raw point cloud-based voxels, its resolution is 20cm. All voxels are recorded in same INTEGER coordinate with offset (336000, 6245250, 20).
- For raw BIM-based voxels, its resolution is 10 cm. Each building is in its own INTEGER coordinate with MINXYZ.
  - For be.xyz, the offset is (336300, 6245507, 25).
  - For blockhouse.xyz, the offset is (336042, 6245613, 27).
  - For dalton.xyz, the offset is (336305, 6245569, 29).
  - For quadrangle.xyz, the offset is (336409, 6245580, 31).
  - For roundhouse.xyz, the offset is (336047, 6245651, 25).
  - For scithe.xyz, the offset is (336325, 6245582, 28).

1.2 VoxelDB Database Schema Design

Four layouts serve as the main tables, and the corresponding two semantic features serve as codelist (to ensure the uniqueness and integrity of semantic information). According to the current data set, we have 5 classes of top-level objects, namely, building, Tree, Green area, road and Terrain. Here we are going to regard the top-level object as a static object, defined as Enumerated types (enum). For detail, I list the data composition as follows:

Object	Direct data sources		Indirectly generated data
	LOD1	LOD4	
building	46 buildings with exact name attribute	6 buildings with exact IFC label and name attribute	94 building footprints extracted from terrain without name attribute and IFC label
tree	793 tree blocks without name attribute and IFC label	N/A	N/A

green area	N/A	N/A	120 green area blocks without name attribute and IFC label
road	N/A	N/A	115 road blocks without name attribute and IFC label
terrain	26947 terrain blocks without name attribute and IFC label	N/A	N/A

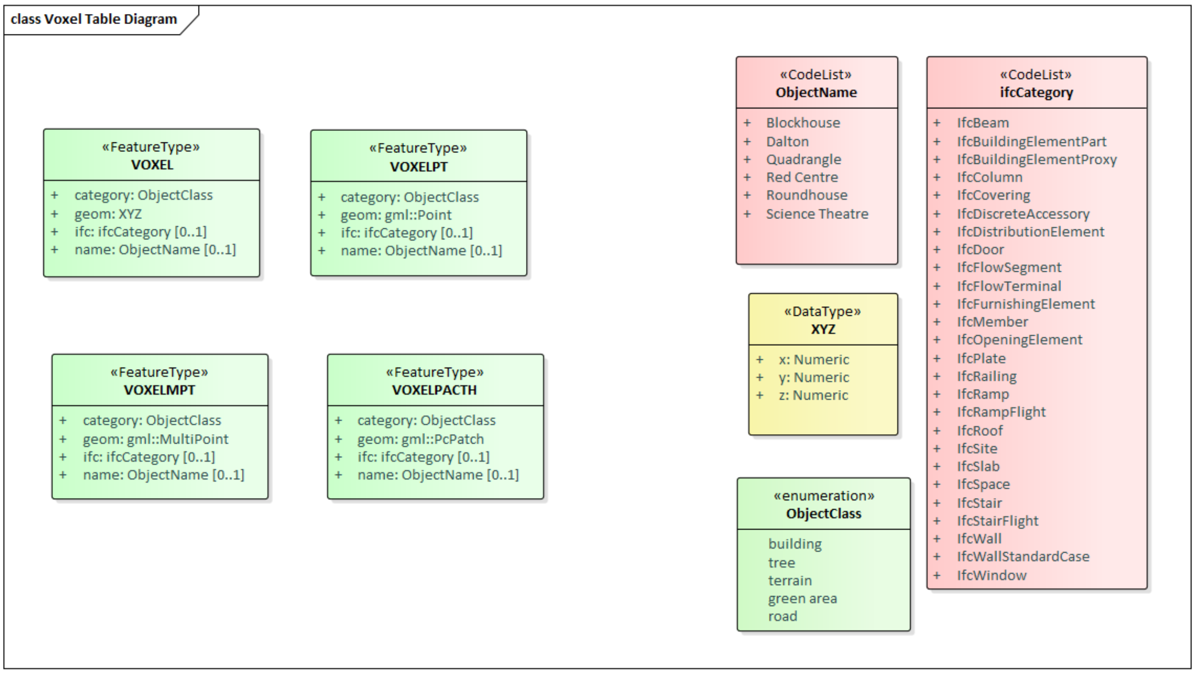

Figure 1-1. Conceptual Model of VoxelDB

1.3 Summary of Four Data Layouts in voxeldb

In this voxeldb, we consider four kinds of data layouts: flat array, point, multipoint, and pcpatch.

Data Layouts/Schemas	**ARRAY**	**POINT**	**MULTIPOINT**	**PCPATCH**
Table Name	voxel	voxelpt	voxelmpt	voxelpatch
Table Size	189 GB	230 GB	4654 MB	474 MB
#records	1,247,611,386	1,247,611,387	28,193	28,193
Geometry	NO	POINT	MULTIPOINT	PCPATCH
Columns	id category ifc name x y z	id category geom ifc name	id category geom ifc name	id category pa ifc name
#index	1	2	2	2
General Index	YES	YES	YES	YES
	x, y, z, category, name, ifc	category, name, ifc	category, name, ifc	category, name, ifc

Spatial Index	NO N/A	YES point geometry	YES multipoint geometry	YES 2D bounds of the patch
Index Size	100 GB	136 GB	5240 kB	5016 kB

1.4 Two Semantic CodeLists in voxeldb (For end-user queries)

1.4.1 ObjectName

The object name given in main table are defined by the codelist "ObjectName". The only semantic information we have so far is the building name.

Code list of *name* attribute (total 48 buildings)		
Rupert Myers-M15	International House-C6	UNSW Village-B10
Barker Street Carpark-N18	Law-F8	Blockhouse-G6
Sam Cracknell Pavilion-H8	Science Theatre-F13	Dalton-F12
Ainsworth Building-J17	Computer Science-K17	Willis Annexe-J18
UNSW Business School-E12	Quadrangle-E15	Electrical Engineering-G17
Rex Vowels Theatre-F17	Robert Webster-G14	Webster Theatres-G15
Red Centre-H13	Newton-J12	Keith Burrows Theatre-J14
Physics Theatre-K14	Old Main-K15	Chemical Sciences-F10
University Terraces-B8	Warrane College-M7	New College-L6
Shalom College-N9	Barker Apartments-N13	House at Pooh Corner-N8
Swimming Pool-B4	UNSW Fitness and Aquatic Centre-B5	Tyree Energy Technologies-H6
Goldstein Hall-D16	UNSW Hall-D14	Fig Tree Theatre-B14d
White House-C15	Roundhouse-E6	Squarehouse-E4
Colombo House-B16	Goldstein College-B17	Fig Tree Hall-B18
Philip Baxter College-D18	Basser College-D17	Old Tote-B15

1.4.2 ifcCategory

The ifc object category given in main table are defined by the codelist "ifcCategory". There are currently 26 IFC objects in our database.

Code list of *ifc* attribute (total 26 IFC objects)		
IfcBeam	IfcBuildingElementPart	IfcBuildingElementProxy
IfcColumn	IfcCovering	IfcDiscreteAccessory
IfcDistributionElement	IfcDoor	IfcFlowSegment
IfcFlowTerminal	IfcFurnishingElement	IfcMember
IfcOpeningElement	IfcPlate	IfcRailing
IfcRamp	IfcRampFlight	IfcSpace
IfcRoof	IfcSite	IfcSlab
IfcStair	IfcStairFlight	IfcWall
IfcWallStandardCase	IfcWindow	

1.5 Functionality and Simple Sample

Semantic query:

- [YES] which category? want to find 'building's? 'terrain's? Use 'category' directly!
- [YES] which building? want to find 'Red Centre'? 'Roundhouse'? Use 'name' and LIKE keyword in your query!

- [YES] which ifc object? want to find ‘IfcDoor’ in certain building or all buildings? Use ‘ifc’ directly!
- [NO] which road object? want to find “UNIVERSITY MALL”?
- [NO] which green area? want to find “quadrangle lawn”?
- [NO] which building footprint? want to find “red centre” building’s footprint in terrain?

Spatial query:

- [YES] all overall spatial queries valid on “multipoint” and “pcpatch”
- [CAN] queries that do geometric splitting, like return door in “red centre” building whose height is larger than 20m? return rooms in ‘red centre’ building in level-4?

1.6 Performance Evaluation

About how the actual query measurements, the queries are executed in a “**cold**” environment, i.e. when no data is in memory/OS or DB cache, and also in a “**hot**” environment, i.e. when data has been already queried and it is in memory/OS or DB cache. However, in this report we mostly present the response times of the “**hot**” queries. *The hot queries via PostgreSQL are shown in parentheses in these tables, the reported time for a hot query is the average of three runs for the query after the very first run, the so-called “**cold**” query.*

Note,

1. through quantitative test (average of three runs), the performance between using “BETWEEN … AND …” and using “>= AND <=” is almost same. For example,

`EXPLAIN ANALYZE` `SELECT * FROM voxel WHERE (x BETWEEN 336100 AND 336150) AND (y BETWEEN 6245400 AND 6245600);`	861,334.565 ms
`EXPLAIN ANALYZE` `SELECT * FROM voxel WHERE x >= 336100 AND x <= 336150 AND y >= 6245400 AND y <= 6245600;`	859,495.741 ms

2. All querying results are in millisecond, i.e., ms.

As for the queries:

A. *Queries that return subset of voxels*

1. Select all voxels in a given MINMAX space (radius). Several options for MINMAX: rectangular space, very long but narrow space, different size space and at different locations (at the beginning, middle and end of the entire voxel space of the campus)

```
– A-1-1: For small rectangle [50*50]
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336100 AND 336150) AND (y BETWEEN 6245300 AND 6245350);

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336100,6245300,336150,6245350,28356)));

EXPLAIN ANALYZE
```

```
SELECT * FROM voxelmpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336100,6245300,336150,6245350,28356)));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE (PC_Intersects(pa, ST_MakeEnvelope(336100,6245300,336150,6245350,28356)));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
1,775,107.963 (1,717,215.492)	280,735.503 (2,856.332)	583.787 (323.011)	27.296 (18.841)

```
-- A-1-2: For large rectangle [200*200]
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336200 AND 336400) AND (y BETWEEN 6245400 AND 6245600);

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336200,6245400,336400,6245600,28356)));

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336200,6245400,336400,6245600,28356)));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE (PC_Intersects(pa, ST_MakeEnvelope(336200,6245400,336400,6245600,28356)));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
866,519.623 (870,766.634)	7,356,650.896 (160,087.365)	22,659.413 (16,529.909)	478.709 (4.406)

```
-- A-1-3: For long, narrow, diagonal rectangle [50*200]
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336100 AND 336150) AND (y BETWEEN 6245400 AND 6245600);

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336100,6245400,336150,6245600,28356)));

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE (ST_Intersects(geom, ST_MakeEnvelope(336100,6245400,336150,6245600,28356)));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE (PC_Intersects(pa, ST_MakeEnvelope(336100,6245400,336150,6245600,28356)));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
1,775,107.963 (1,717,215.492)	280,735.503 (2,856.332)	583.787 (323.011)	27.296 (18.841)
866,519.623 (870,766.634)	7,356,650.896 (160,087.365)	22,659.413 (16,529.909)	478.709 (4.406)
849,299.853 (851,131.126)	1,184,770.067 (24,818.530)	18,746.538 (14,884.024)	47.163 (0.788)

870,430.593 (863,186.358)	488,798.495 (9,541.339)	13,922.886 (10,642.721)	889.276 (5.473)
857,826.172 (861,149.265)	7,501,617.02 (201,697.697)	27,006.244 (20,192.778)	1,103.469 (24.454)

2. Select all voxels in a given MINMAX/radius from a voxel (x,y,z)

```
-- A-2-1: For small cycle at (336300,6245500), radius 20m
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336300-20.0 AND 336300+20.0)
AND (y BETWEEN 6245500-20.0 AND 6245500+20.0)
AND SQRT(POW(x-336300, 2) + POW(y-6245500, 2)) < 20;

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),20,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),20,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE (PC_Intersects(pa, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),20,'quad_segs=8')));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
870,430.593 (863,186.358)	488,798.495 (9,541.339)	13,922.886 (10,642.721)	889.276 (5.473)

```
-- A-2-2: For large cycle at (336300,6245500), radius 100m
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336300-20.0 AND 336300+20.0)
AND (y BETWEEN 6245500-20.0 AND 6245500+20.0)
AND SQRT(POW(x-336300, 2) + POW(y-6245500, 2)) < 100;

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE (PC_Intersects(pa, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
857,826.172 (861,149.265)	7,501,617.02 (201,697.697)	27,006.244 (20,192.778)	1,103.469 (24.454)

3. Select all voxels that have a specific attribute

```
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE category='tree';

EXPLAIN ANALYZE
```

```sql
SELECT * FROM voxelpt WHERE category='tree';

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE category='tree';

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE category='tree';
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
859,403.914 (860,772.063)	647,474.552 (659,147.888)	12.155 (0.525)	8.146 (0.301)

4. Combination of 3 and 4: e.g. all trees in a given MINMAX/radius around a given location (x,y,z)

```sql
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336300-20.0 AND 336300+20.0)
AND (y BETWEEN 6245500-20.0 AND 6245500+20.0)
AND SQRT(POW(x-336300, 2) + POW(y-6245500, 2)) < 100 AND category='tree';

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE category='tree' AND (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelmpt WHERE category='tree' AND (ST_Intersects(geom, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT * FROM voxelpatch WHERE category='tree' AND (PC_Intersects(pa, ST_Buffer(ST_SetSRID(ST_MakePoint(336300,6245500),28356),100,'quad_segs=8')));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
25,910.208 (14323.357)	96,867.322 (10,282.892)	1,160.399 (766.901)	3.651 (2.624)

5. Select all the information for a specific voxel (x,y,z)

```sql
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE x=336307.7 AND y=6245536.3 AND z=36.6;

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE ST_X(geom)=336307.7 AND ST_Y(geom)=6245536.3 AND ST_Z(geom)=36.6;

EXPLAIN ANALYZE
SELECT ST_X(POINT_geom) AS x, ST_Y(POINT_geom) AS y, ST_Z(POINT_geom) AS z, category, ifc, name
FROM (
    SELECT (ST_Dump(geom)).geom AS POINT_geom, category, ifc, name FROM voxelmpt
) AS temp
WHERE ST_X(POINT_geom)=336307.7 AND ST_Y(POINT_geom)=6245536.3 AND ST_Z(POINT_geom)=36.6;

EXPLAIN ANALYZE
```

```
SELECT PC_Get(PCPOINT_geom, 'X') AS x, PC_Get(PCPOINT_geom, 'Y') AS y, PC_Get(PCPOINT_geom, 'Z') AS z, category, ifc, name
FROM (
    SELECT PC_Explode(pa) AS PCPOINT_geom, category, ifc, name FROM voxelpatch
) AS temp
WHERE PC_Get(PCPOINT_geom, 'X')=336307.7 AND PC_Get(PCPOINT_geom, 'Y')=6245536.3 AND PC_Get(PCPOINT_geom, 'Z')=36.6;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
855,694.652 (924,176.186)	966,600.728 (650,465.419)	598,136.347 (589,097.675)	778,559.791 (778,725.953)

6. Select MAX or MIN on the basis of value (for example ‘ the highest point of DTM of a building X)

```
EXPLAIN ANALYZE
SELECT MAX(z)
FROM voxel
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;

EXPLAIN ANALYZE
SELECT ST_ZMax(geom)
FROM voxelpt
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;

EXPLAIN ANALYZE
SELECT ST_ZMax(geom)
FROM voxelmpt
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;

EXPLAIN ANALYZE
SELECT PC_PatchMax(pa, 'z')
FROM voxelpatch
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
857,504.083 (853,644.877)	643,823.215 (643257.833)	1,994.907 (1,583.951)	58.56 (0.791)

7. Select a specific object (e.g. Building X, DTM, Vegetation)

```
EXPLAIN ANALYZE
SELECT *
FROM voxel WHERE category='building' AND name LIKE 'Red%' AND ifc='IfcStair';

EXPLAIN ANALYZE
SELECT *
FROM voxelpt WHERE category='building' AND name LIKE 'Red%' AND ifc='IfcStair';

EXPLAIN ANALYZE
SELECT *
FROM voxelmpt WHERE category='building' AND name LIKE 'Red%' AND ifc='IfcStair';
```

```

EXPLAIN ANALYZE
SELECT *
FROM voxelpatch WHERE category='building' AND name LIKE 'Red%' AND ifc='IfcStair';
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
197,539.499 (39339.156)	123,949.187 (30,815.353)	88.209 (0.328)	0.325 (0.274)
79,183.874 (41510.013)	31331.503 (32001.347)	2.964 (0.156)	0.315 (0.116)

8. Select all objects (e.g. buildings) with a given attribute (e.g. have indoor)

```
EXPLAIN ANALYZE
SELECT *
FROM voxel WHERE category='building' AND ifc='IfcDoor';

EXPLAIN ANALYZE
SELECT *
FROM voxelpt WHERE category='building' AND ifc='IfcDoor';

EXPLAIN ANALYZE
SELECT *
FROM voxelmpt WHERE category='building' AND ifc='IfcDoor';

EXPLAIN ANALYZE
SELECT *
FROM voxelpatch WHERE category='building' AND ifc='IfcDoor';
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
79,183.874 (41510.013)	31331.503 (32001.347)	2.964 (0.156)	0.315 (0.116)

B. Some more complicated selections

1. Give all neighbours of a voxel (x,y,z)

```
EXPLAIN ANALYZE
SELECT * FROM voxel WHERE (x BETWEEN 336307.7-0.2 AND 336307.7+0.2)
AND (y BETWEEN 6245536.3-0.2 AND 6245536.3+0.2) AND (z BETWEEN 36.6-0.2 AND 36.6+0.2)
ORDER BY POW(x-336307.7, 2) + POW(y-6245536.3, 2) + POW(z-36.6, 2)
LIMIT 100;

EXPLAIN ANALYZE
SELECT * FROM voxelpt WHERE (ST_X(geom) BETWEEN 336307.7-0.2 AND 336307.7+0.2)
AND (ST_Y(geom) BETWEEN 6245536.3-0.2 AND 6245536.3+0.2) AND (ST_Z(geom) BETWEEN 36.6-0.2 AND 36.6+0.2)
ORDER BY POW(ST_X(geom)-336307.7, 2) + POW(ST_Y(geom)-6245536.3, 2) + POW(ST_Z(geom)-36.6, 2)
LIMIT 100;

```

```
EXPLAIN ANALYZE
SELECT *
FROM (
    SELECT (ST_Dump(geom)).geom AS POINT_geom, category, ifc, name FROM voxelmpt
) AS temp
WHERE (ST_X(POINT_geom) BETWEEN 336307.7-0.2 AND 336307.7+0.2)
AND (ST_Y(POINT_geom) BETWEEN 6245536.3-0.2 AND 6245536.3+0.2) AND (ST_Z(POINT_geom) BETWEEN 36.6-0.2 AND 36.6+0.2)
ORDER BY POW(ST_X(POINT_geom)-336307.7, 2) + POW(ST_Y(POINT_geom)-6245536.3, 2) + POW(ST_Z(POINT_geom)-36.6, 2)
LIMIT 100;

EXPLAIN ANALYZE
SELECT *
FROM (
    SELECT PC_Explode(pa) AS PCPOINT_geom, category, ifc, name FROM voxelpatch

) AS temp
WHERE (PC_Get(PCPOINT_geom, 'X') BETWEEN 336307.7-0.2 AND 336307.7+0.2)
AND (PC_Get(PCPOINT_geom, 'Y') BETWEEN 6245536.3-0.2 AND 6245536.3+0.2) AND (PC_Get(PCPOINT_geom, 'Z') BETWEEN 36.6-0.2 AND 36.6+0.2)
ORDER BY POW(PC_Get(PCPOINT_geom, 'X')-336307.7, 2) + POW(PC_Get(PCPOINT_geom, 'Y')-6245536.3, 2) + POW(PC_Get(PCPOINT_geom, 'Z')-36.6, 2)
LIMIT 100;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
858,781.698 (857,409.365)	642,644.831 (643,330.872)	615,713.452 (603,627.066)	1,228,798.352 (1,229,903.46)

2. Give the buffer of an object

```
-- Returns a geometry/geography that represents all trees
-- whose distance from 'Red Centre' is less than or equal to 20m

EXPLAIN ANALYZE
SELECT V2.* FROM voxelmpt V1, voxelmpt V2
WHERE V1.category='building' AND v2.category='tree' AND V1.name LIKE 'Red%' AND V1.ifc IS NULL
AND (ST_Intersects(V2.geom, ST_Buffer(ST_Envelope(V1.geom),20,'quad_segs=8')));

EXPLAIN ANALYZE
SELECT V2.* FROM voxelpatch V1, voxelpatch V2
WHERE V1.category='building' AND v2.category='tree' AND V1.name LIKE 'Red%' AND V1.ifc IS NULL
AND (PC_Intersects(V2.pa, ST_Buffer(PC_EnvelopeGeometry(V1.pa),20,'quad_segs=8')));
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
N/A	N/A	24,255.72 (23,876.509)	51.862 (4.123)

C. *Queries that return a result of computation*

1. Give the volume of an object (objects)

```sql
-- volume of 'Red Centre'
CREATE EXTENSION postgis_sfcgal;


EXPLAIN ANALYZE
SELECT (ST_XMax(box)-ST_XMin(box))*(ST_YMax(box)-ST_YMin(box))*(ST_ZMax(box)-
ST_ZMin(box)) AS volume
FROM (SELECT ST_3DExtent(geom) AS box FROM voxelmpt
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL) AS tmp;

EXPLAIN ANALYZE
SELECT (PC_PatchMax(pa, 'x')-PC_PatchMin(pa, 'x'))*(PC_PatchMax(pa, 'y')-
PC_PatchMin(pa, 'y'))*(PC_PatchMax(pa, 'z')-PC_PatchMin(pa, 'z')) AS volume
FROM voxelpatch
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
N/A	N/A	1,338.614 (1347.161)	1.177 (1.091)

2. Give the area of the footprint of an object

```sql
EXPLAIN ANALYZE
SELECT ST_Area(ST_Envelope(geom))
FROM voxelmpt
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;

EXPLAIN ANALYZE
SELECT ST_Area(PC_EnvelopeGeometry(pa))
FROM voxelpatch
WHERE category='building' AND name LIKE 'Red%' AND ifc IS NULL;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
N/A	N/A	1,229.905 (1,179.873)	0.85 (0.357)

3. Give the distance between voxel 1 and voxel 2

```sql
EXPLAIN ANALYZE
SELECT SQRT(POW(336057.2-336307.7, 2) + POW(6245674.5-
6245536.3, 2) + POW(35.6-36.6, 2)) FROM voxel;

EXPLAIN ANALYZE
SELECT SQRT(POW(336057.2-336307.7, 2) + POW(6245674.5-
6245536.3, 2) + POW(35.6-36.6, 2)) FROM voxelpt;

EXPLAIN ANALYZE
SELECT SQRT(POW(336057.2-336307.7, 2) + POW(6245674.5-
6245536.3, 2) + POW(35.6-36.6, 2)) FROM voxelmpt;

EXPLAIN ANALYZE
SELECT SQRT(POW(336057.2-336307.7, 2) + POW(6245674.5-
6245536.3, 2) + POW(35.6-36.6, 2)) FROM voxelpatch;
```

voxel (FLAT ARRAY)	voxelpt (POINT GEOMETRY)	voxelmpt (MULTIPOINT GEOMETRY)	voxelpatch (PCPATCH)
825,877.422 (827,786.899)	638,521.516 (636,008.835)	217.338 (26.686)	59.991 (4.591)

Table. Summary of the above queries.

Querying time in millisecond (ms)				
Query	**voxel (FLAT ARRAY)**	**voxelpt (POINT GEOMETRY)**	**voxelmpt (MULTIPOINT GEOMETRY)**	**voxelpatch (PCPATCH)**
A-1-1	1,775,107.963 (1,717,215.492)	280,735.503 (2,856.332)	583.787 (323.011)	27.296 (18.841)
A-1-2	866,519.623 (870,766.634)	7,356,650.896 (160,087.365)	22,659.413 (16,529.909)	478.709 (4.406)
A-1-3	849,299.853 (851,131.126)	1,184,770.067 (24,818.530)	18,746.538 (14,884.024)	47.163 (0.788)
A-2-1	870,430.593 (863,186.358)	488,798.495 (9,541.339)	13,922.886 (10,642.721)	889.276 (5.473)
A-2-2	857,826.172 (861,149.265)	7,501,617.02 (201,697.697)	27,006.244 (20,192.778)	1,103.469 (24.454)
A-3	859,403.914 (860,772.063)	647,474.552 (659,147.888)	12.155 (0.525)	8.146 (0.301)
A-4	25,910.208 (14323.357)	96,867.322 (10,282.892)	1,160.399 (766.901)	3.651 (2.624)
A-5	855,694.652 (924,176.186)	966,600.728 (650,465.419)	598,136.347 (589,097.675)	778,559.791 (778,725.953)
A-6	857,504.083 (853,644.877)	643,823.215 (643257.833)	1,994.907 (1,583.951)	58.56 (0.791)
A-7	197,539.499 (39339.156)	123,949.187 (30,815.353)	88.209 (0.328)	0.325 (0.274)
A-8	79,183.874 (41510.013)	31331.503 (32001.347)	2.964 (0.156)	0.315 (0.116)
B-1	858,781.698 (857,409.365)	642,644.831 (643,330.872)	615,713.452 (603,627.066)	1,228,798.352 (1,229,903.46)
B-2	N/A	N/A	24,255.72 (23,876.509)	51.862 (4.123)
C-1	N/A	N/A	1,338.614 (1347.161)	1.177 (1.091)
C-2	N/A	N/A	1,229.905 (1,179.873)	0.85 (0.357)
C-3	825,877.422 (827,786.899)	638,521.516 (636,008.835)	217.338 (26.686)	59.991 (4.591)

References

[1] Introduction to UML, https://github.com/ISO-TC211/UML-Best-Practices/wiki/Introduction-to-UML

[2] Gröger, G., Kolbe, T.H., Nagel, C. and Häfele, K.H., 2012. OGC city geography markup language (CityGML) encoding standard.

[3] Pointcloud, https://github.com/pgpointcloud/pointcloud

[4] Li, W., Zlatanova, S., Diakite, A.A., Aleksandrov, M. and Yan, J., 2020. Towards Integrating Heterogeneous Data: A Spatial DBMS Solution from a CRC-LCL Project in Australia. ISPRS International Journal of Geo-Information, 9(2), p.63.