

# CityJSON in combination with MongoDB, PostgreSQL and GraphQL

Karin Staring - 4952510

P5 presentation

---

Stelios Vitalis

Linda van den Brink

Balázs Dukai

Willem Korthals Altes

```
<gml:boundedBy>
  <gml:Envelope srsName="urn:ogc:def:crs:EPSG::28992" srsDimension="3">
    <gml:lowerCorner>90409.32 435440.44 0.0</gml:lowerCorner>
    <gml:upperCorner>91453.879 436089.946 41.158</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="{3E04FD3E-CDF4-4EC0-88BA-645257958409}">
    <gen:doubleAttribute name="TerrainHeight">
      <gen:value>1.97</gen:value>
    </gen:doubleAttribute>
    <gen:stringAttribute name="bron_tex">
      <gen:value>UltraCAM-X 10cm juni 2008</gen:value>
    </gen:stringAttribute>
    <gen:stringAttribute name="voll_tex">
      <gen:value>complete</gen:value>
    </gen:stringAttribute>
    <gen:stringAttribute name="bron_geo">
      <gen:value>Lidar 15-30 punten - nov.</gen:value>
    </gen:stringAttribute>
    <gen:stringAttribute name="status">
      <gen:value>1</gen:value>
    </gen:stringAttribute>
  </bldg:Building>
  <bldg:boundedBy>
    <bldg:RoofSurface gml:id="UUID_77d73b95-62e0-4f3a-833a-000000000000">
      <bldg:lod2MultiSurface>
        <gml:MultiSurface>
          <gml:surfaceMember>
            <gml:Polygon gml:id="UUID_89ca3b34-039a-456a-8000-000000000000">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList srsDimension="3">
                    91081.975 435795.152 11.776 91081.3850000000
                  </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
        <gml:surfaceMember>
          <gml:Polygon gml:id="UUID_7f0b8bfb-1aec-4c8c-adc5-088000000000">
```

# CityGML (XML

# CityJSON (JSON)

```
"type": "CityJSON",
"version": "1.0",
"metadata": {
    "referenceSystem": "urn:ogc:def:crs:EPSG::28992",
    "geographicalExtent": [
        90409.32,
        435440.44,
        0,
        91453.879,
        36089.946,
        158
    ]
}

N (JSON)
{
    "id": "-45EE-836A-1DB9FEF06E04": {
        "Building",
        "routes": {
            "TerrainHeight": 1.37,
            "bron_tex": "UltraCAM-X 10cm juni 2008",
            "voll_tex": "complete",
            "bron_geo": "Lidar 15-30 punten - nov. 2008",
            "status": "1"
        },
        "geometry": [
            {
                "type": "MultiSurface",
                "boundaries": [
                    [
                        [
                            [
                                6244,
```

Building (UUID\_8e26c0bb-68b9-4f52-a5cb-cc054871207f)

Building (UUID\_0d9c0284-a859-4193-9528-a832f575fd63)

Building (UUID\_a41d83a6-c179-4bf3-af3b-6228e6df6692)

Building (UUID\_a2cdcbc2b-37ff-4113-bb44-eadcc6b883c2)

Building (UUID\_9d9b51cd-fdd1-40f9-ba4a-dfeec40d6e44)

Building (UUID\_cc578dcc-449d-48eb-8a2f-8f01469a5ac5)

Building (UUID\_69a2aef9-a6e6-4755-90b2-0bdb3bdb7006)

Building (UUID\_141905c5-1692-4ca4-ae3e-4348633a0d0d)

Building (UUID\_0f7a2b6b-32e5-4730-bad4-0bb470727f7c)

Building (UUID\_93fad353-113a-4e7c-b9a9-132e77c46b89)

Building (UUID\_87adc5ec-5a07-471a-9bf9-53bad975ea56)

Building (UUID\_56682be2-c7bc-4471-900c-9cbbc59b0236)

Building (UUID\_b26f11d2-b79d-439f-9f04-b96e430fff77)

Building (UUID\_16ce0bfa-8c47-4164-a058-d1945e69f54b)

Building (UUID\_5d55beb8-c19d-4160-b265-fcc70f656659)

Building (UUID\_0ec53f1c-89bb-458e-8d4b-3e7d2c646958)

Building (UUID\_f8ddebdf-7d35-4a00-8fca-2dfaf31d3b3d)

Building (UUID\_ab5ad8f8-1fdc-4bcd-97ab-dc947c4b8819)

Building (UUID\_53dacc14-19f7-4a77-baae-622541c4ca7e)

Building (UUID\_4999bcb3-67cd-4564-b6bd-f190faa2e1f1)

Building (UUID\_1a624476-c817-4f26-ad50-04350826582b)

Building (UUID\_8f6c21ca-f52e-4ce8-ac1f-5eda30832c34)

Building (UUID\_d889fad3-c551-4814-b63c-e9c26b878868)

Building (UUID\_1d956563-f6bb-47bf-9d5b-70766d407a5b)

Building (UUID\_59eb9491-33b2-4a73-8ac5-9f774ab6ae59)

Building (UUID\_f07a8b86-4ab8-45fe-bfa2-b3f6e0b0a2ae)

Building (UUID\_a116e149-85bc-4126-82a0-0568c588cf65)

Building (UUID\_b834c976-f989-45b0-a4b2-266ee6dfbd2)

Building (UUID\_749d3067-7303-4341-a969-66daddef02f2)

Attribute	Value
OS	1141
TILE	3367218.391959628 5806639.061027913
DFIHO	0
DFORM	0
FOKHO	31320
FOKHW	5807393170
FOKDM	2269003260

A 3D perspective view of a terrain model. The terrain is composed of numerous small red rectangular blocks representing building footprints or walls. A single large, semi-transparent yellow polyhedron is positioned in the center-right area, partially overlapping the terrain. The background shows a dark blue gradient, suggesting a sky or distant landscape.

3

# MongoDB = NoSQL/document database

The screenshot shows the MongoDB Compass interface. On the left sidebar, under the 'Local' section, there are 5 DBs and 7 Collections. The 'CityJSON' collection is selected, showing its sub-collections: CityObjects, metadata, transform, admin, config, local, and test. The 'CityObjects' sub-collection is currently active, displaying four document entries. Each entry is a building object with fields: \_id, type, attributes, geometry, and metadata\_id. The first document's \_id is '{00138BE2-F4EB-4B9E-A6F0-77EECB4E9D56}', type is 'Building', attributes is an object, geometry is an array, and metadata\_id is 'metadata\_delfshaven'. The second document has a similar structure with \_id '{0015085B-C2E8-43AE-98DF-6A0E807A9DAA}'. The third and fourth documents have \_ids '{0031458C-0F2F-40B4-B170-90D6305F9F78}' and '{008F0D4C-BFDE-4419-B7B4-D4C489234FA4}' respectively.

_id	type	attributes	geometry	metadata_id
{00138BE2-F4EB-4B9E-A6F0-77EECB4E9D56}	Building	Object	Array	metadata_delfshaven
{0015085B-C2E8-43AE-98DF-6A0E807A9DAA}	Building	Object	Array	metadata_delfshaven
{0031458C-0F2F-40B4-B170-90D6305F9F78}	Building	Object	Array	metadata_delfshaven
{008F0D4C-BFDE-4419-B7B4-D4C489234FA4}	Building	Object	Array	metadata_delfshaven

# PostgreSQL = Relational database

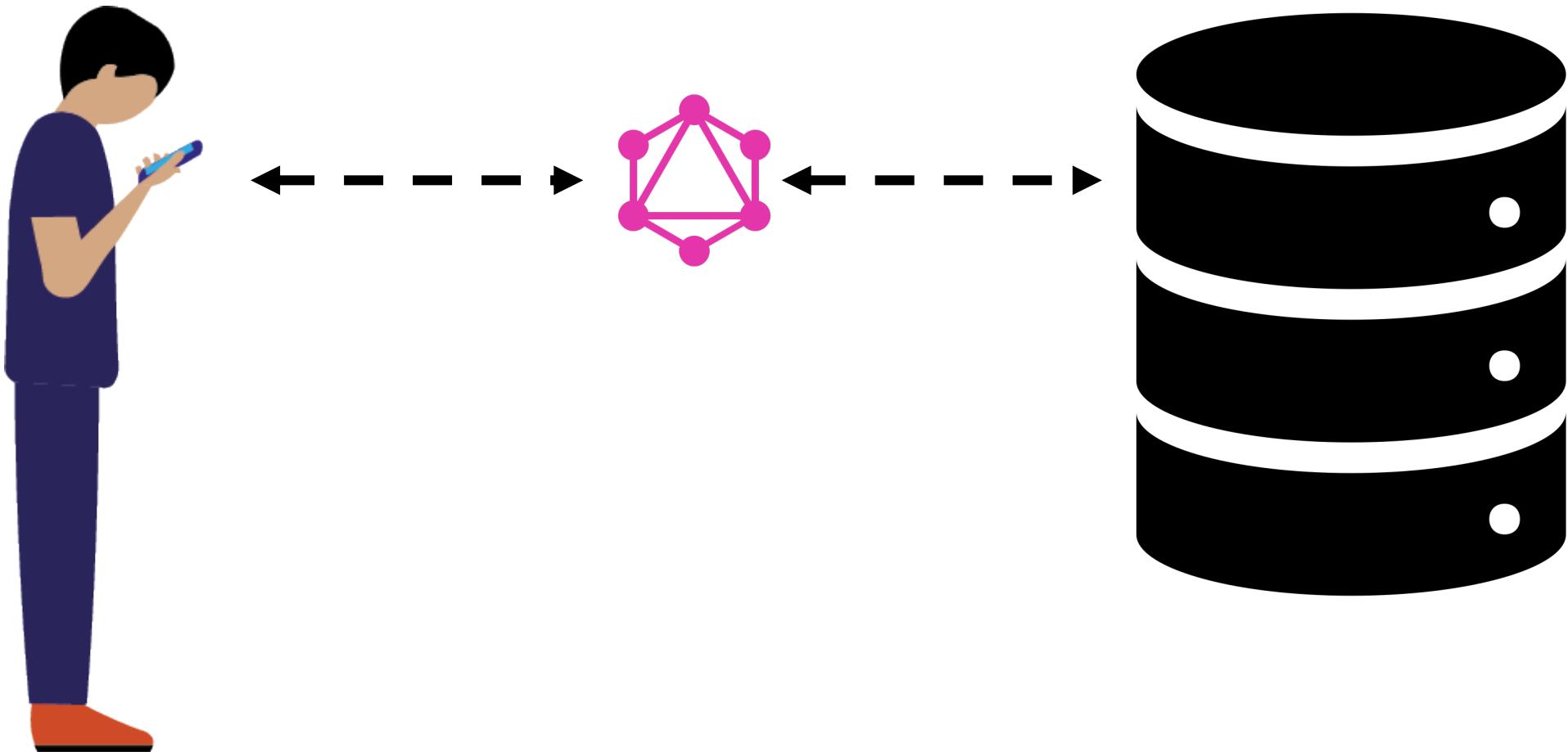
The screenshot shows the pgAdmin 4 application interface. The left sidebar displays a hierarchical list of databases and their objects. The main area features a toolbar at the top with various icons for file operations, followed by a dashboard panel and a query editor window.

**Servers (1)**

- PostgreSQL
  - Databases (5)
    - delfshaven\_3DCityDB
    - denhaag\_3DCityDB
  - insertdb
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Schemas (2)
      - citysondb
        - Collations
        - Domains
        - FTS Configurations
        - FTS Dictionaries
        - FTS Parsers
        - FTS Templates
        - Foreign Tables
        - Functions
        - Materialized Views
        - Procedures
        - Sequences
      - Tables (7)
        - city\_object
        - geometries
        - metadata
        - parents\_children
        - semantic\_surface
        - surfaces
        - transform
      - Trigger Functions
      - Types
      - Views
      - public
- postgres
  - Casts
  - Catalogs
  - Event Triggers
  - Extensions
  - Foreign Data Wrappers
  - Languages
  - Schemas

- psotmdash\_3DCityDB
- Casts
- Catalogs
- Event Triggers
- Extensions
- Foreign Data Wrappers
- Languages
- Schemas

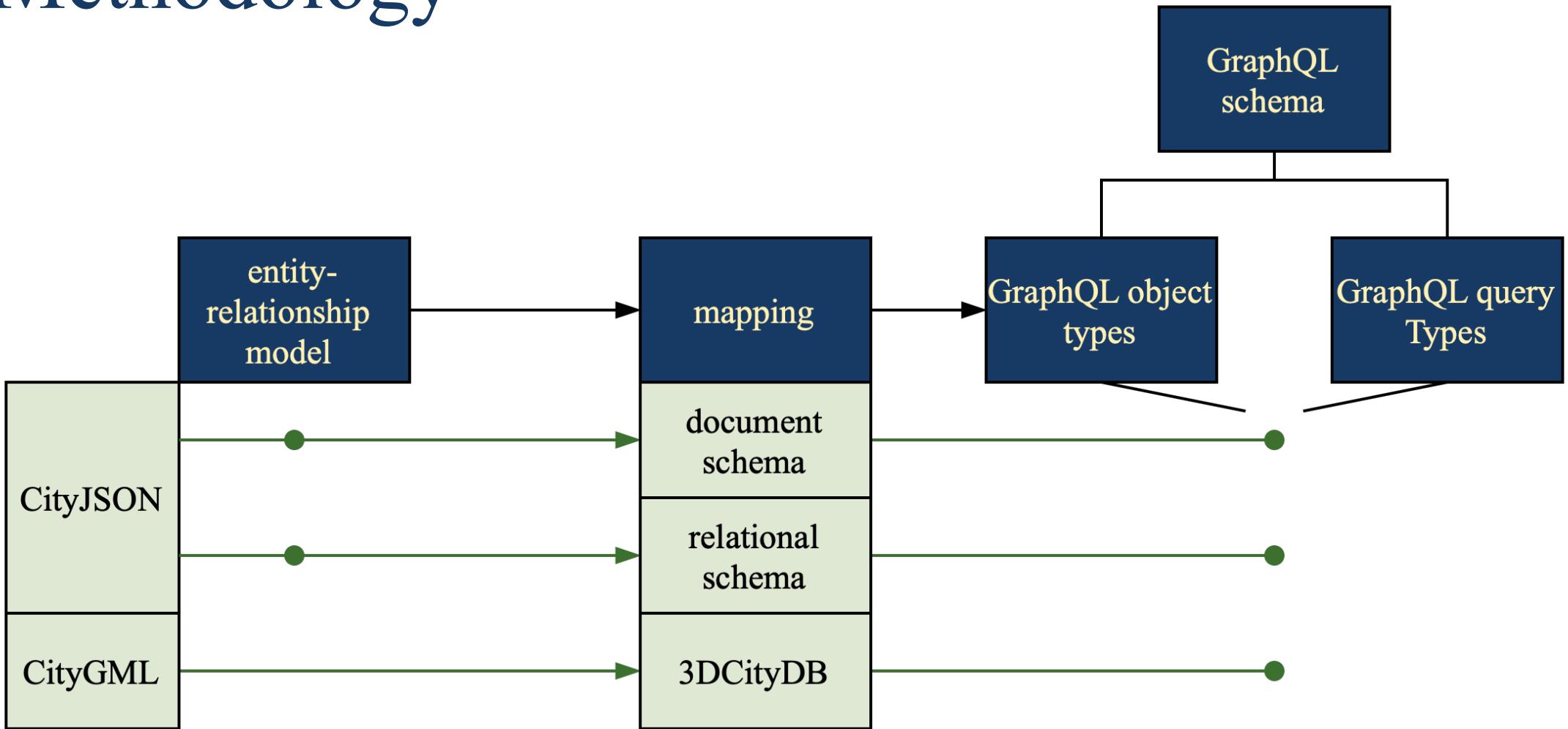
# Use case



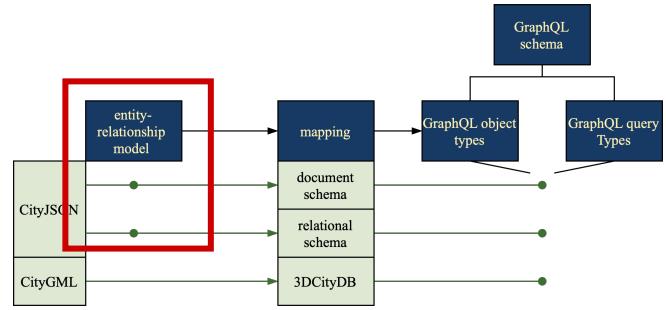
# Research

How suitable are MongoDB and PostgreSQL for the storage  
and querying of CityJSON using GraphQL?

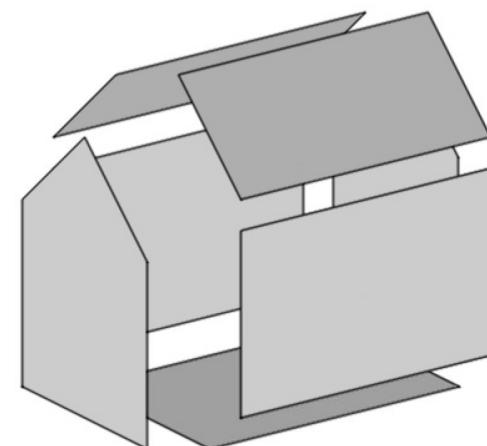
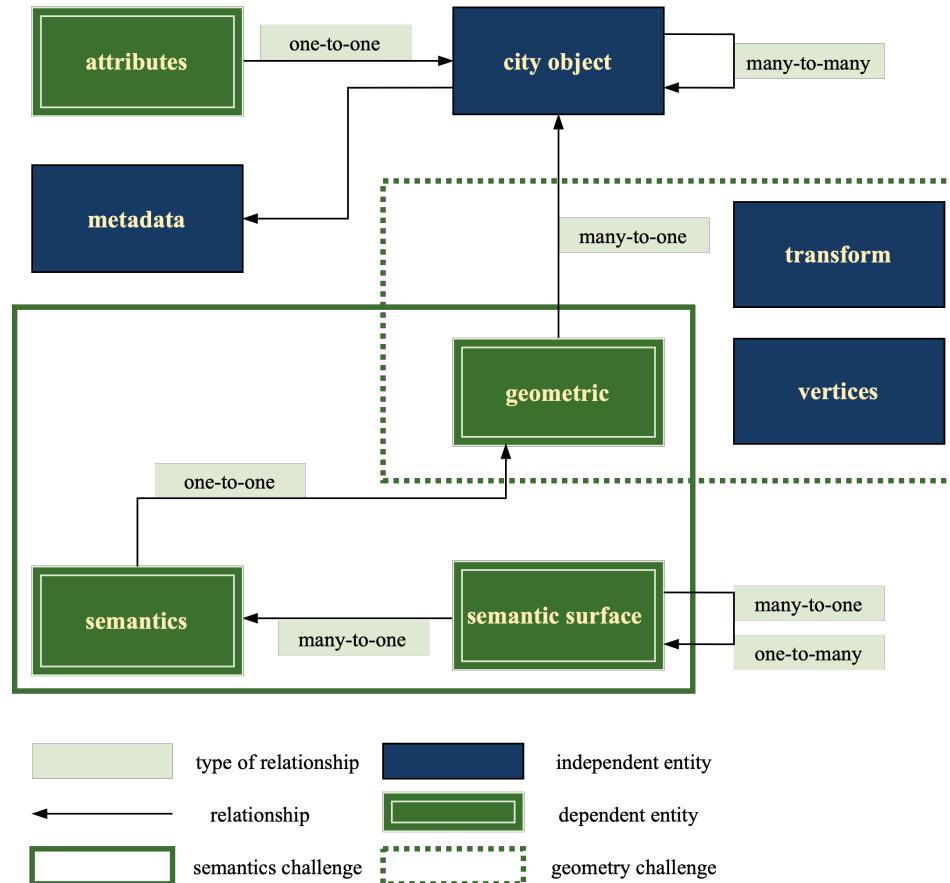
# Methodology



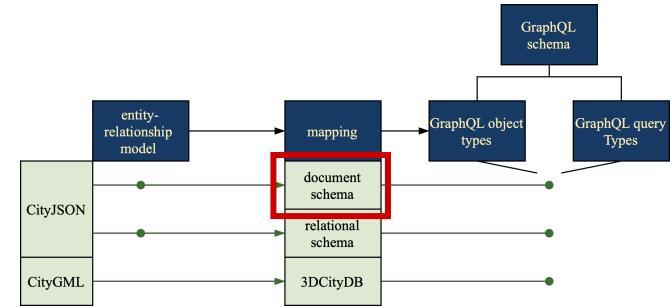
# Entity-relationship model



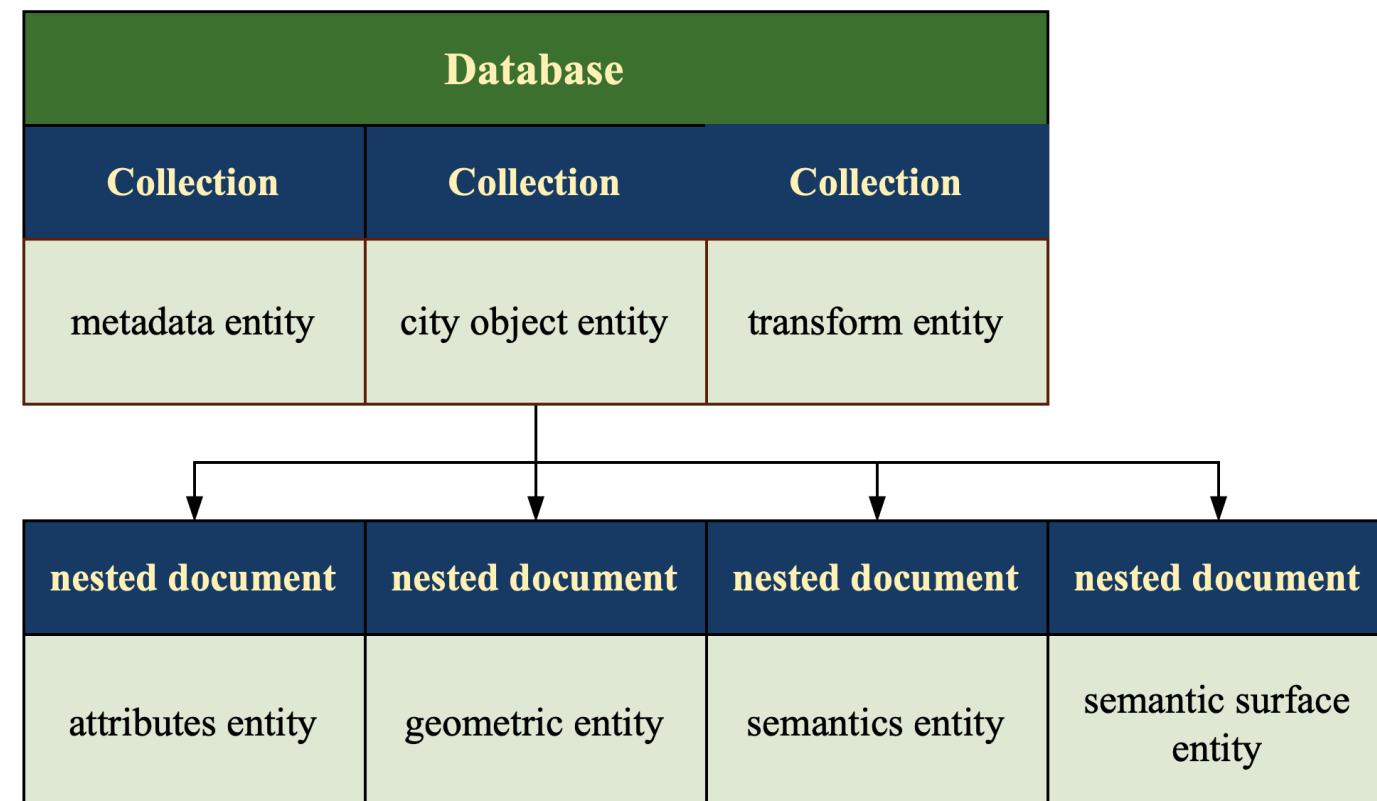
- Dependent/ independent entities
- Many-to-many relationship
- Geometry challenge
- Semantics challenge
- JSON



# Document schema



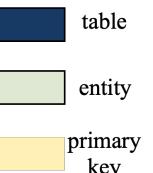
- Dependent/ independent entities
- Many-to-many relationship
- Geometry challenge
- Semantics challenge

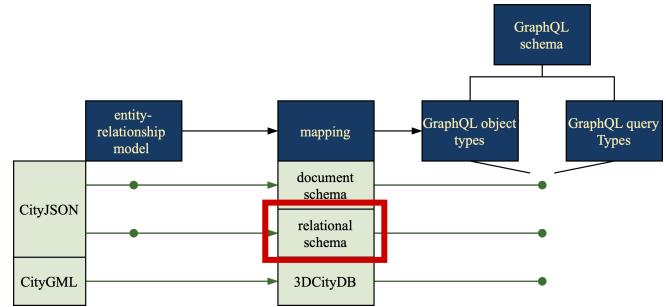


# Relational schema

- Dependent/ independent entities
- Many-to-many relationship
- Geometry challenge
- Semantics challenge

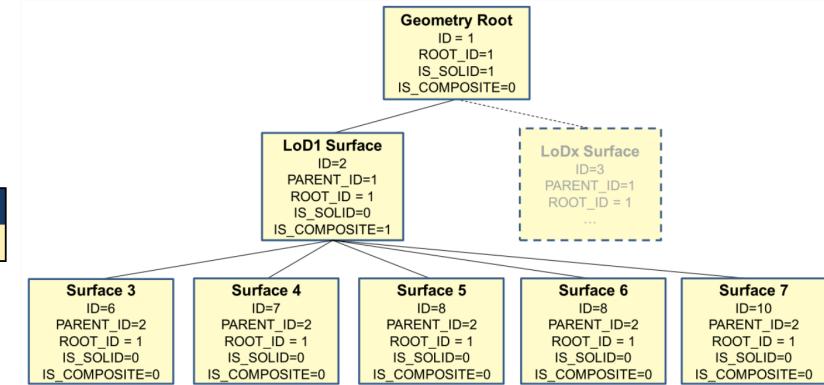
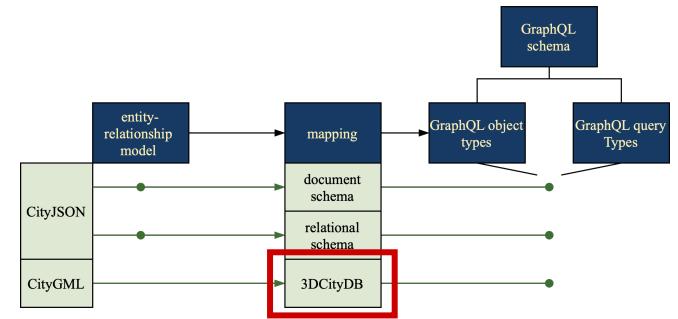
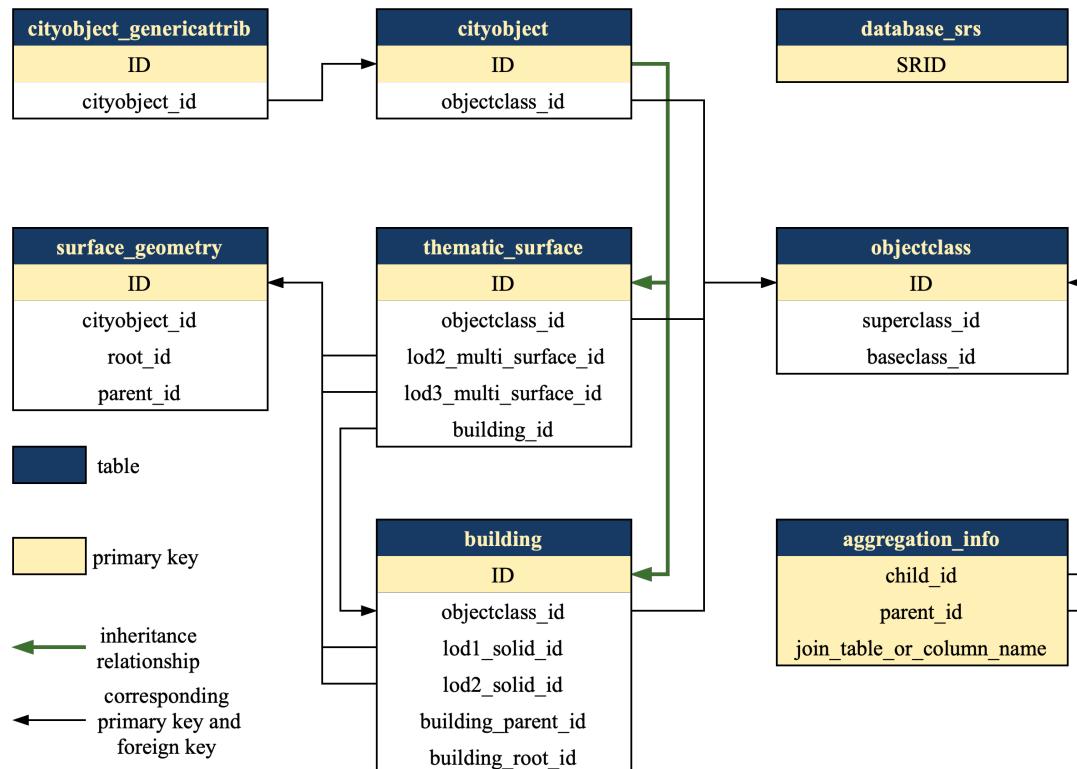
Database			
metadata table	city_object table	transform table	parents_children table
metadata entity	city object entity	transform entity	
id [text] object [jsonb]	id [text] object [jsonb] attributes [jsonb] metadata_id <i>...extra geometry</i>	id [text] object [jsonb]	parents_id children_id
geometry table	surfaces table	semantic_surface table	
geometric entity	semantics entity	semantic surface entity	
id [sequence] object [jsonb] city_object_id	id [sequence] geometry [polygons] solid_num [int] shell_num [int] surface_num [int] geometry_id semantic_surface_id city_object_id	id [sequence] object [jsonb] city_object_id children parent	The semantics entity is unnecessary, because the geometry is split into surfaces. These surfaces are directly linked to the objects of the semantic surface entity.


  
 table  
 entity  
 primary key



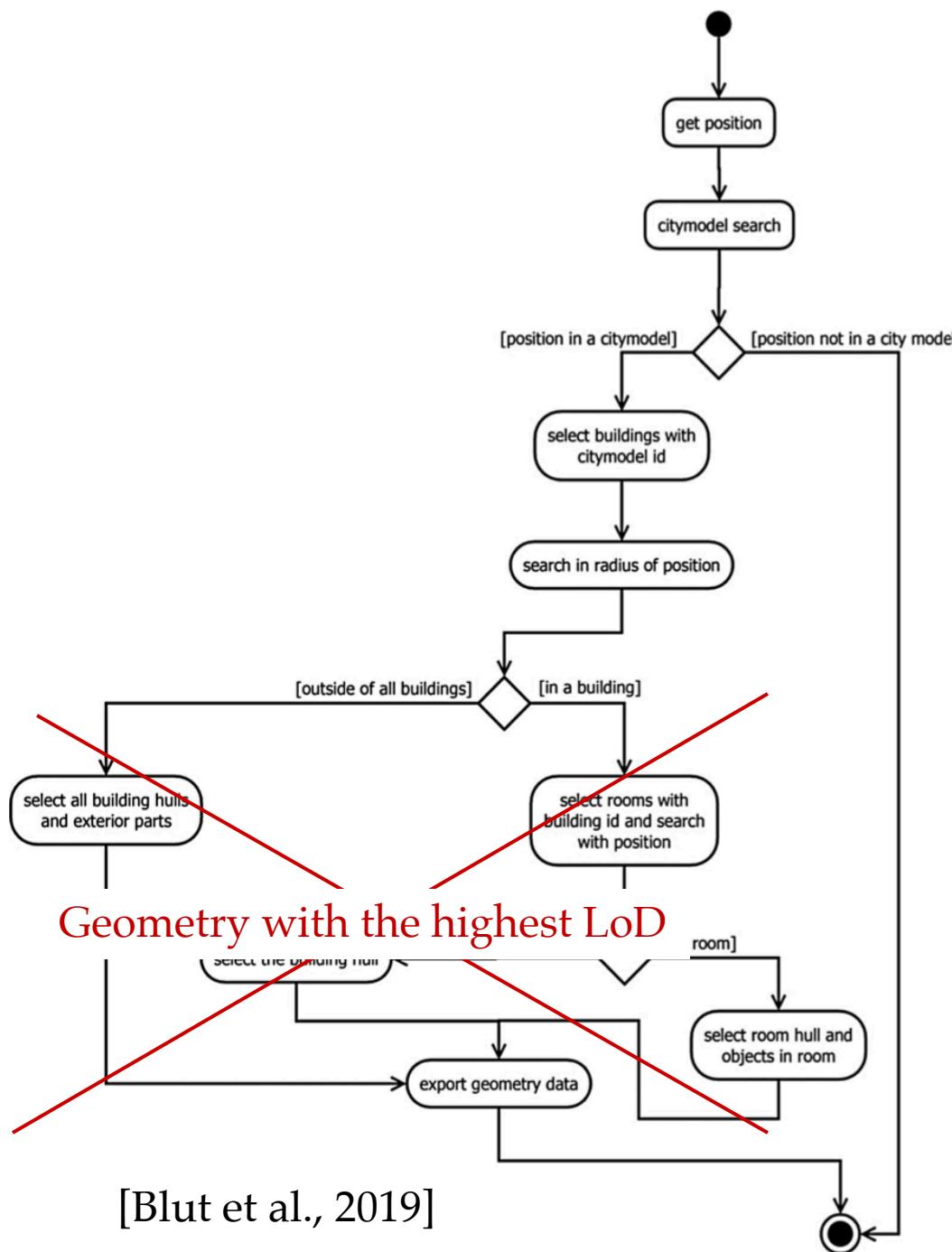
# 3DCityDB

- Inheritance relationship
- Generic attributes
- Geometries



[Kolbe, 2019]

# Selection process

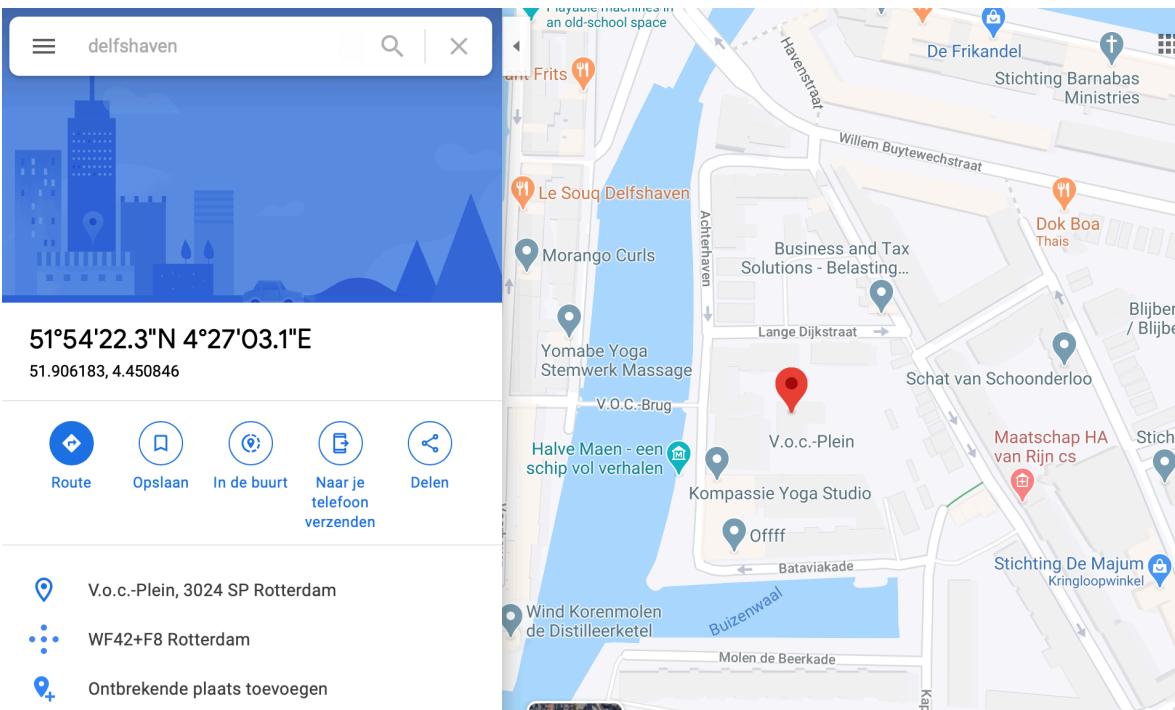


# The 7 queries

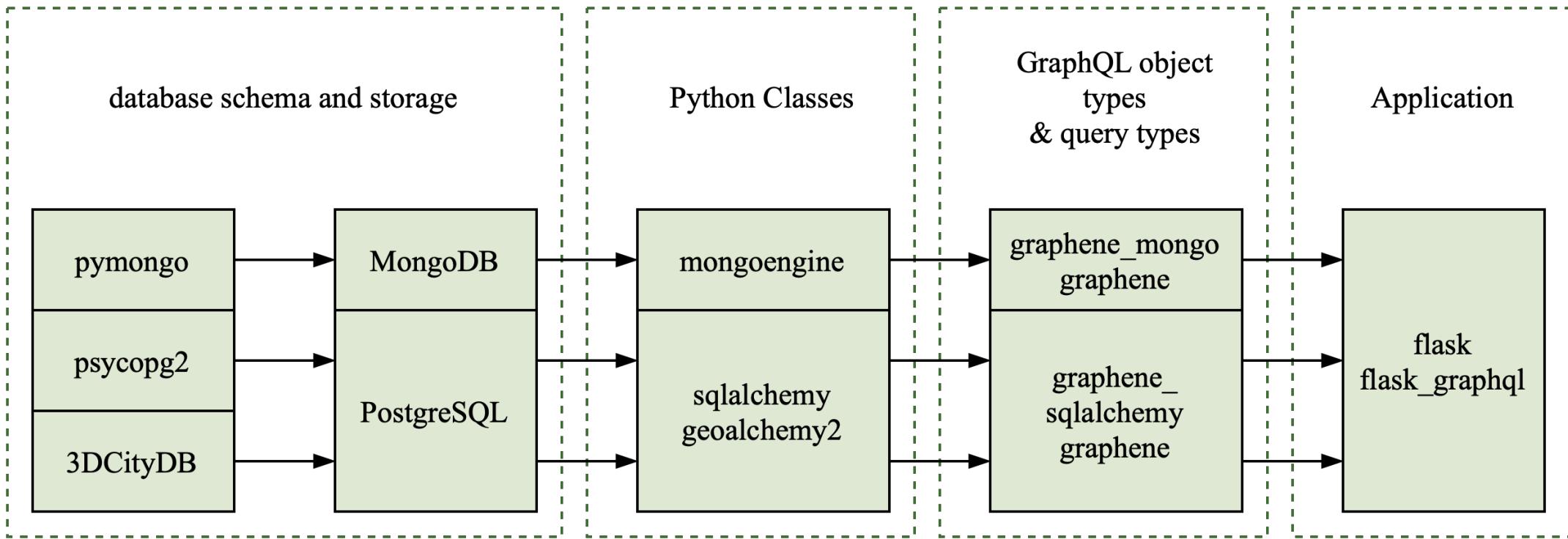
query number and name	argument	operations	filter	returned data
1: location	position with the fields lat, long and alt			location with the fields latitude, longitude and altitude
2: citymodel	position with the fields lat, long and alt	intersection		city model ID
3: radius100	position with the fields lat, long and alt	within 100 meter	city object type = Building	city object IDs
4: inside	position with the fields lat, long and alt	intersection	city object type = Building	city object ID
5: maxlod	city object ID			ID of the geometry with the highest LoD
6a: cityobjects	city object ID		ID-based	return the ID of the city object
6b: cityobjects	city object ID		ID-based	return the ID and attributes of the city object
6c: cityobjects	city object ID		ID-based	return the ID and geometries of the city object
7: surfaces	surface ID or other		ID-based or other	return the surface with the related semantic surface object

# Data

<b>Denhaag</b>	2.6 MB	EPSG:4715	parent-child relationship	
<b>Delfshaven</b>	1.4 MB	EPSG: 28992		
<b>Potsdam</b>	15.7 MB	EPSG: 25833	multiple LoDs	presentLoDs



# Software

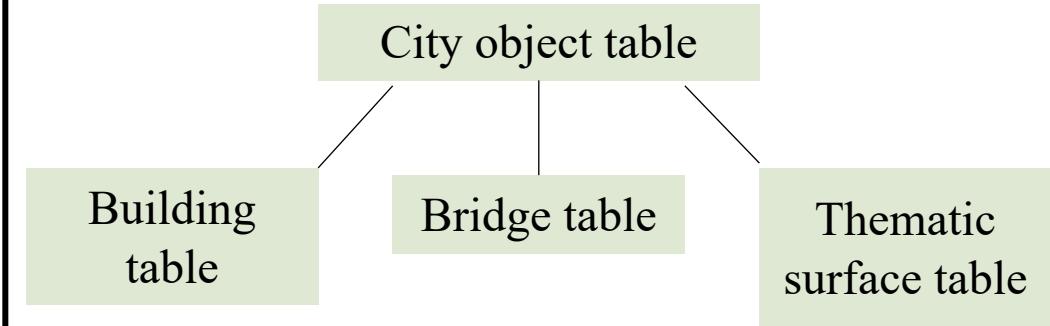


## CityJSON in PostgreSQL

	<b>id</b> [PK] text	<b>object</b> jsonb	<b>attributes</b> jsonb
1	UUID_93f...	{"type": "Building..."	{"Geomtype": 1, "creation..."
2	UUID_c98...	{"type": "Building", ...	{"class": "BB01", "Region" ...
3	UUID_a4a...	{"type": "Building..."	{"Geomtype": 1, "creation..."
4	UUID_ba0...	{"type": "Building", ...	{"class": "BB07", "Region" ...

```
{ cityobjects {  
    id  
}}
```

## CityGML in PostgreSQL with 3DCityDB



```
{ cityobjects {  
    building {  
        id  
    }}}
```

## CityJSON in PostgreSQL

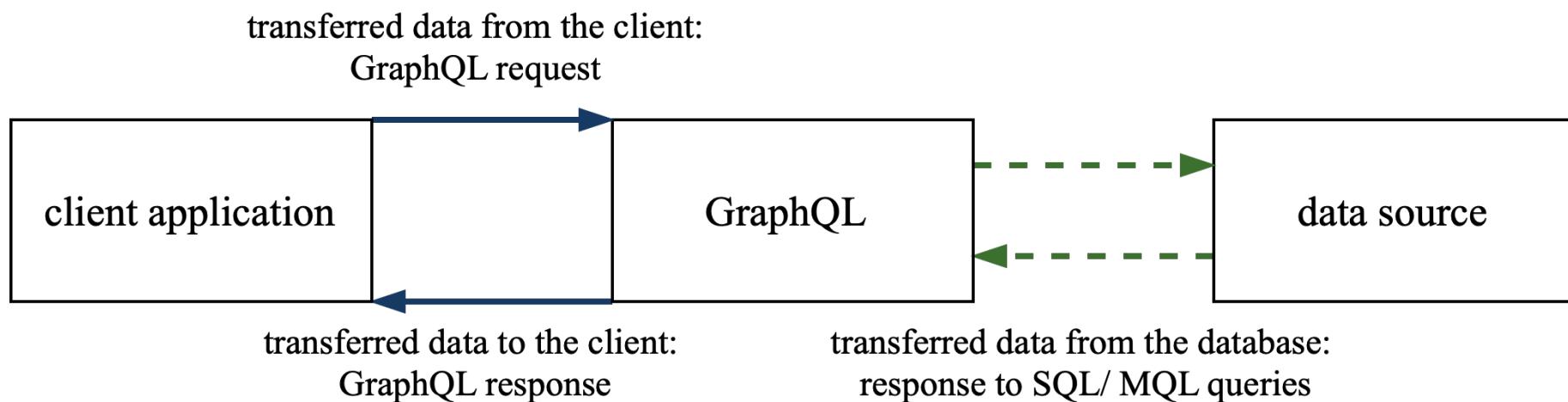
## CityJSON in MongoDB

	"presentLoDs": {"2.0": 145865}
one city object – multiple tables	one city object – one document
reference transformations	no reference transformations
Maximum field size = 1GB	Maximum document size = 16MB

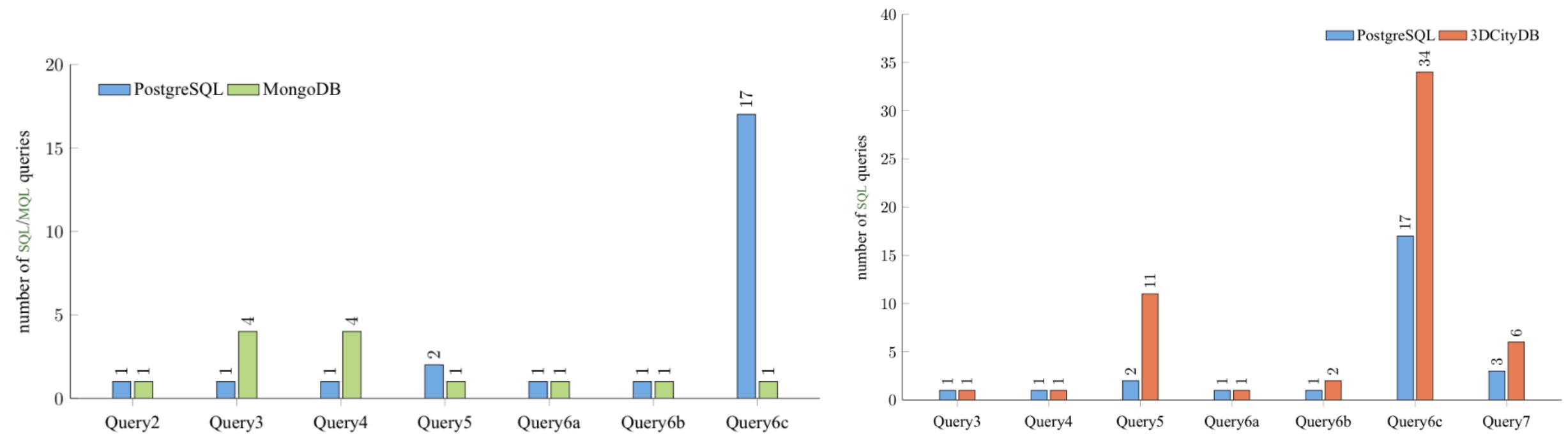
```
MultiSurface:  
boundaries = ListField(ListField(ListField(IntField())))  
MultiSolid:  
boundaries = ListField(ListField(ListField(ListField(ListField(IntField())))))
```

# Experiments

- Retrieval times
- Request and response sizes
- Over-fetching



# Results



# Results

- More queries lead to higher retrieval times
  - Possible due to ORM/ODM → not using joins
- More overfetching for MongoDB and 3DCityDB than PostgreSQL
  - Possible due to ORM/ODM → not selecting fields

# How suitable are MongoDB and PostgreSQL for the storage and querying of CityJSON using GraphQL?

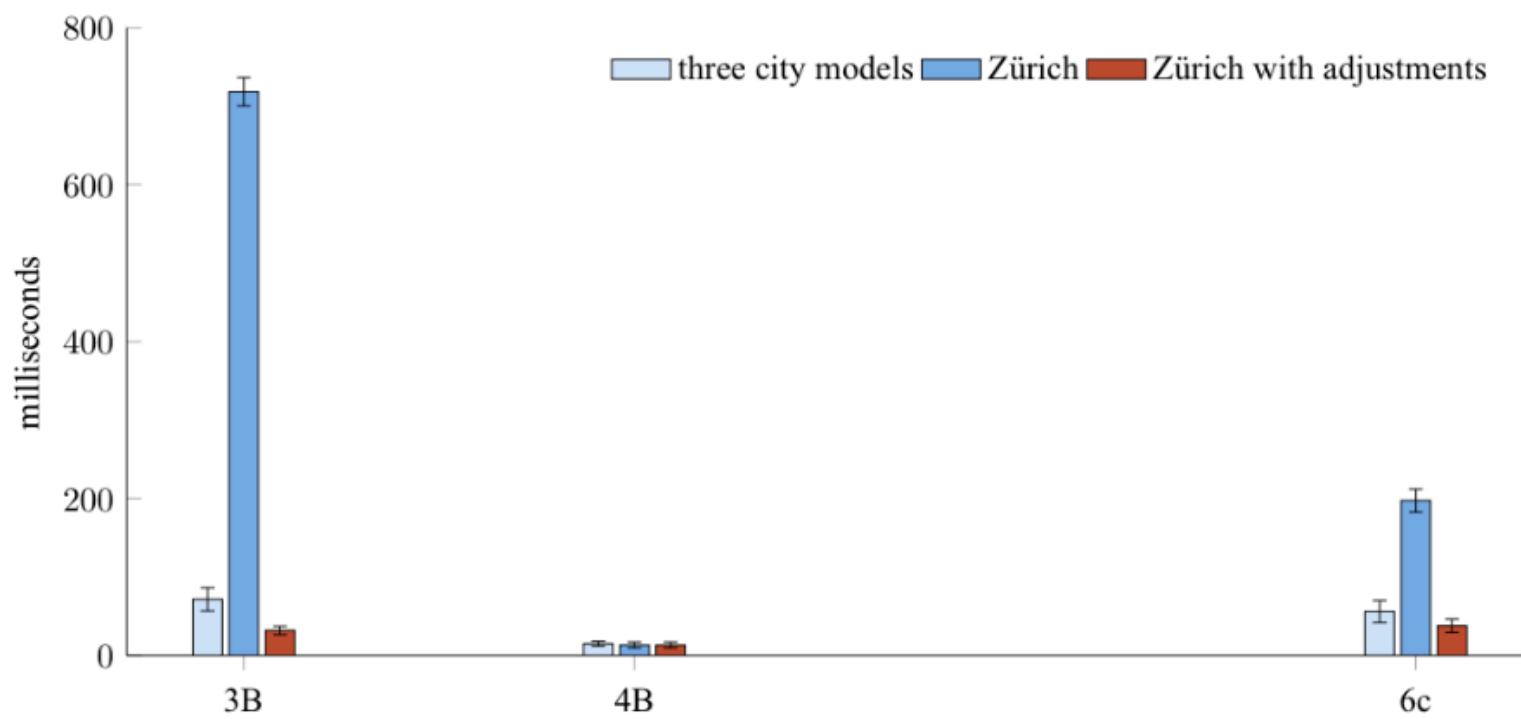
- No real signs yet that MongoDB and PostgreSQL are not suitable
  - Vertices list
  - Attribute presentLoDs
  - reference transformations
- Integration of JSON
  - Attributes are mapped more flexibly → less tables → less queries/joins → lower retrieval times
  - Support for fields with varying data types
    - + support to map the hierarchy of arrays
    - - GraphQL is less flexible
  - Querying a JSON attribute -→ higher retrieval times = not investigated

# Recommendations

- Implement the SQL/MQL queries as efficient as possible without being connected to GraphQL first
  - + better understanding database performance
  - + better understanding abilities GraphQL
- A more general understanding of the suitability for all use cases could be provided with a framework that tests more types of queries.
  - Aggregate queries
  - Queries on GeoJSON objects in MongoDB
  - Queries on JSON attributes in PostgreSQL and MongoDB

# Recommendations

	Three datasets together	Zürich
Size (MB)	19.7	292.8
number of objects	6258	198699



# Questions?

# Spatial queries per implementation

Use one city model instead of multiple

PostgreSQL	MongoDB	
supports many reference systems geometrical spatial operations on 2D and 3D geometries geographical spatial operations on 2D geometries		supports GeoJSON objects in 2D with WGS84 as global reference system
CityGML in PostgreSQL with 3DCityDB	CityJSON in PostgreSQL	CityJSON in MongoDB
envelope transformed in the query (in the database)	convex hull transformed in the query (in the database)	global convex hull transformed before query
index on envelope	additional indexes: index on convex hull index on global convex hull	

Store convex hull as GeoJSON object