ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XII-4/W2-2026
10th International Conference on Smart Data and Smart Cities (SDSC), 29 September–1 October 2026, Sofia, Bulgaria

From Planning Question to Spatial Database: A Geospatial Data Scheme-Aware Importer for Urban Digital Twins

Iván Cárdenas-León[1], Mila Koeva[1], Pirouz Nourian[1], Karin Pfeffer[1]

[1]Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente, Enschede, NL
— i.l.cardenasleon@utwente.nl, m.v.koeva@utwente.nl, p.nourian@utwente.nl, k.pfeffer@utwente.nl

Keywords: Geospatial Data Interoperability, Spatial Database, Planning Support Systems, Open Urban Platforms, Digital Twins, Urban Planning.

Abstract

Urban Digital Twins (UDTs) promise cross-sector, evidence-based planning support, yet a foundational challenge persists: ingesting heterogeneous, multi-source geospatial data into a unified, queryable spatial database. This paper presents a manual methodology for translating spatial planning questions and an open-source web-based ingestion interface for configuring spatial databases. The approach begins by formalizing the planning question as a Directed Acyclic Graph (DAG), in which each node represents a measurable concept along with its associated data attributes. These nodes are then translated directly into Django ORM (Object-Relational Mapping) models within a PostgreSQL/PostGIS database, creating a schema-aware foundation for data ingestion. Built on this foundation, a four-tier web application comprising a GeoDjango backend, a React frontend, a Django REST API, and a TiTiler raster tile server was designed to enable non-expert users to upload, field-map, and validate vector, raster, and tabular geospatial data through a single interface. Coordinate reference system reprojection, schema verification, and validation are handled automatically, facilitating the configuration of spatial databases. This model-driven, ORM-based ingestion workflow aims to reduce schema management complexity, compared to manually specified SQL-based approaches, while keeping the application and database layers in sync. Future work will need to evaluate the usability and time reduction of the current tool, and extend the platform to support scenario simulation, LLM-assisted data retrieval from external sources, and real-time connectivity with Smart City sensors via a unified database.

1. Introduction

Urban environments are rapidly evolving into interconnected, data-driven ecosystems in which the complexity of city management increasingly exceeds the capacity of traditional planning tools (Alvi et al., 2025). Urban Digital Twins (UDTs) have emerged as one approach to addressing this challenge. As an extension of the Smart City movement, the aim is to provide dynamic representations of cities that, in theory, enable real-time monitoring, scenario simulation, and evidence-based decision-making across sectors such as mobility, energy, and public health (Allan et al., 2025; Ketzler et al., 2020; Lei et al., 2023).

The Digital Twin concept originated in aerospace engineering as a method for managing assets in outer space (Shafto et al., 2010). It later evolved in the manufacturing industry, where it is defined as a continuously updated digital counterpart to a physical system that informs real-world operations (Grieves, 2015; Kritzinger et al., 2018). When translated to the urban scale, Urban Digital Twins are commonly described as integrated platforms that combine real-time data streams, spatial modeling, and simulation to support monitoring, stakeholder collaboration, sustainable urban planning, and decision-making processes (Caprari et al., 2022; Dembski et al., 2020; White et al., 2021). A well-designed UDT has the potential to generate cross-sector insights, enable evidence-based policymaking, facilitate interdisciplinary collaboration, break down data silos, and enhance stakeholder engagement (Patel et al., 2026).

Despite this potential, and the growing research and investment in UDT platforms, a persistent bottleneck remains: the ingestion and integration of heterogeneous, multi-source, and multi-dimensional geospatial data into a structured and queryable database (Alvi et al., 2025; Jeddoub et al., 2023, 2025). Differences in data sources, spatial and temporal resolutions, storage and exchange formats, data-collection objectives, and update frequencies create substantial heterogeneity, hindering synchronization and completeness in UDT systems (Wysocki et al., 2026). Schema mismatches, inconsistent attribute naming, complex data configurations, and broader interoperability issues continue to limit the operational maturity of Digital Twin platforms (Patel et al., 2026; Quek et al., 2023). Moreover, a cross-sectoral impact analysis process requires transforming diverse datasets into a structured spatial database that supports optimized simulations within an Urban Digital Twin (Wang et al., 2026). Addressing these issues through middleware and schema-matching approaches introduces additional operational complexity but remains essential for ensuring synchronization, consistency, and long-term interoperability (Patel et al., 2026).

Previous research has acknowledged this challenge, yet existing systems typically prioritize visualization and querying over comprehensive data ingestion. For instance, Fargas Jr. et al. (2024) and Kipkemboi et al. (2023) both developed spatial platforms built on GeoDjango and PostGIS stacks, demonstrating the viability of web-based interfaces for multi-source urban data. However, both systems presuppose that data are already formatted and externally hosted. The former relies on GeoServer, Open Data Cube, and remote WMS (Web-Map Service) endpoints for data access, while the latter constructs its schema directly via SQL (Structured Query Language) scripts, which limits maintainability and scalability. Neither system provides a mechanism for raw file ingestion, coordinate reference system normalization, or schema mapping from heterogeneous sources. While tools such as GeoServer and QGIS offer partial solutions for specific data types, many UDT implementations still lack

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XII-4-W2-2026-33-2026

a schema-aware, data-model-driven ingestion pipeline that can handle vector, raster, and tabular data through a unified interface (Mandal et al., 2026; Wang et al., 2026). As a result, adding new datasets often still requires developer intervention, limiting the transferability of such frameworks to new urban contexts and data environments.

Geospatial ETL (Extract - Transform - Load) tools offer partial solutions but introduce constraints of their own. Schema-matching platforms such as HALE Studio (Horak et al., 2011; wetransform GmbH & halestudio contributors, 2024) and the commercial FME (Feature Manipulation Engine) are designed for batch harmonization toward standards-compliant outputs (CityGML, INSPIRE, WFS [Web Feature Service]) but operate as standalone processes external to the spatial planning application. GeoServer (Aime et al., 2026) and GDAL (Geospatial Data Abstraction Library) address data serving and format conversion, respectively, but presuppose that data is already structured and externally hosted.

In response to these challenges, this paper addresses the following research question: ***How can a planning question be systematically translated into a configured spatial database that supports cross-sectoral Urban Digital Twin development, without requiring expert SQL knowledge or manual schema definition?*** To answer this, we present a manual methodology and an open-source web application that formalize planning questions as Directed Acyclic Graphs, derive relational database schemas from those graphs, and provide a schema-aware ingestion interface for heterogeneous geospatial data. The primary contribution aimed in this paper is, not the individual components, but the methodological pipeline that connects a planning question to a queryable spatial database through a model-driven workflow accessible to non-expert users.

The system presented in this paper builds on the GeoDjango + PostGIS + COG (Cloud Optimized GeoTiff) established stack (Fargas Jr. et al., 2024; Kipkemboi et al., 2023; White et al., 2023) and extends it with a DAG-driven, model-derived ingestion pipeline, embedding schema definition, format handling, and CRS (Coordinate Reference System) normalization directly within the application layer, without dependence on external ETL tools or pre-formatted data sources.

2. From Planning Questions to Directed Acyclic Graphs

The main goal of any digital planning system, including UDTs, should be to answer questions that support informed decision-making (Potts & Webb, 2023). Planning questions can be broadly classified into two operational modes: *monitoring*, which interrogates current system states (e.g., *"How affordable is water in this city?"*, *"What proportion of network water is lost to leakage?"*), and *scenario planning*, which evaluates conditional futures (e.g., *"How affordable will water be in 2050 under projected climate trajectories?"*). This distinction is not merely semantic: it governs the spatial structure and causal depth required of the underlying data model.

Several formalisms exist for representing the conceptual structure of a planning question. UML (Unified Modeling Language) class diagrams capture structural relationships well but do not encode directionality or functional dependency between variables. Ontological representations, while expressive, introduce reasoning complexity that exceeds the requirements of schema generation. ER (Entity-Relationship) diagrams, though closely related to relational schemas, do not distinguish between associative and causal relationships, nor do they enforce acyclicity. A DAG, by contrast, offers two properties that are particularly valuable here: *directionality*, which makes explicit that some variables are inputs to others rather than peers; and *acyclicity*, which guarantees an unambiguous topological ordering of node evaluation, a prerequisite for deriving a migration-safe database schema. We therefore propose that any planning question can be operationalized by decomposing its measurement into a DAG as a causal inference framework (Pearl, 2009), in which directed edges encode functional dependencies rather than mere associations, and the absence of cycles reflects the irreversibility of most physical and socioeconomic processes relevant to urban planning. This decomposition is driven by the iterative application of the question *"How is this measured?"* until atomic, empirically observable variables are reached.

Each node in the DAG is classified along two complementary dimensions. The first is its *role* within the planning system, as defined by the DPSIR framework (Driving forces, Pressures, States, Impacts, Responses) (European Environment Agency, 1999). Driving force nodes represent boundary conditions or exogenous scenario parameters (e.g., climate change). Pressure and State nodes represent directly measurable variables with a defined temporal resolution. Impact nodes represent derived indicators relevant to the planning question. Response nodes relate to intervention options and policy changes arising from observed impacts and updated states.

The second classification dimension is the node *type*. Spatial nodes represent physically or socioeconomically measurable variables (e.g., precipitation volume at a gauge station, or household water expenditure). Aggregation nodes represent derived quantities computed from one or more spatial nodes through arithmetic operations (e.g., total operational expenditure). Input nodes represent assumptions or parameters that can be specified to propagate computations within the DAG. In addition to these classifications, each spatial node requires a defined spatial reference, temporal resolution, and measurement unit — properties determined by the planning question that propagate downward through the graph. A monitoring question may operate at the neighbourhood scale with daily reporting, while a scenario-planning question targeting 2050 outcomes can tolerate coarser resolution. These attributes need not be homogeneous across the DAG: upstream nodes representing physical measurements typically demand finer resolution than downstream socioeconomic indicators, and such mismatches must be resolved through appropriate aggregation or interpolation between connected nodes.

Given these definitions, the decomposition procedure can be made transferable through the following steps: (1) state the planning question in the form *"What is the current/future state of X?"*; (2) for each variable, ask *"How is this measured?"* and record all direct antecedents; (3) classify each node by DPSIR role and type (Spatial, Aggregation, or Input); (4) assign spatial reference, temporal resolution, and measurement unit to each node; (5) verify acyclicity before proceeding; (6) document the rationale for each directed edge explicitly, as edge direction encodes domain assumptions that are not recoverable from the graph structure alone.

Once a planning question has been decomposed into a validated DAG and its node attributes formalized, the graph structure contains sufficient information to generate a relational database schema: nodes map to tables, node attributes map to columns,

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XII-4-W2-2026-33-2026

and directed edges map to foreign key constraints or condition selections (Driver to Pressure). In the current implementation, this translation is performed manually during system architecture setup (Section 3).

As an example, Figure 1 illustrates this approach for the water tariff affordability question. Following Chapagain et al. (2022), affordability is operationalized as total annual operating revenue normalized by population served. Operational costs are further decomposed into energy consumption, volumetric extraction, and network maintenance (Alegre et al., 2017), while available water supply is modeled as a function of precipitation and infiltration, both of which are themselves sensitive to climate change. The DAG thus captures not only the immediate data dependencies but also the broader socio-environmental drivers relevant to long-term planning. Each node maps directly to a relational database table: for instance, the precipitation node requires a point geometry, a temporal attribute, and a rainfall volume measure, while the water extraction node requires spatial coordinates, pumping flow rate, operational duration, energy consumption rate, and extraction depth. All spatially referenced nodes maintain a foreign key relationship with their respective administrative boundaries, enabling spatial aggregation and cross-domain queries.

3. Proposed Framework and System Architecture

The framework for importing data follows a four-tier architecture comprising (1) a PostgreSQL/PostGIS database and Python-based processing backend; (2) a React-based frontend interface; (3) a Django REST API layer (Django Software Foundation, 2025); and (4) a raster tiling and visualization layer. Figure 2 summarizes the workflow across these tiers. The design aims to improve data interoperability, validate required attributes, and reduce manual schema engineering through ORM-based model definitions. Its primary goal is to enable non-expert users to import data into a common schema.

3.1 Database Layer

The first layer consists of a PostgreSQL database extended with the PostGIS and PostGIS-Raster spatial extensions, providing a centralized database that can be migrated to other systems and easily accessed through the desktop version of any GIS software. The geospatial features are managed through GeoDjango ORM, which maps Python model Classes to PostgreSQL Table columns and the PostGIS geometry column, using the same structure detected when creating the DAG. These are called Data models and belong to a Django Application that we have structured to correspond to planning sectors or thematic domains, such as water supply, urban heat, weather, or housing. Rather than manually defining and populating database tables through SQL scripts, the schema can be specified in Python model classes and managed through the Django administration interface.

For instance, Algorithm 1 shows a basic structure of table attributes, including Foreign Keys related to a water source and user destination location; self-calculated length of the pipe measured in km based on the geometry of the object; and self-calculated transformation of pipe diameter from mm to inch and vice versa. Additionally, help strings can be configured to clarify the units of measurement for each attribute. This help text can later be connected to the frontend view for updating data in the database.

Once several sectors and models are structured, it is possible to create a `MODEL REGISTRY` list with all the models of the project, which, in turn, can be organized into `VECTOR REGISTRY` and `RASTER REGISTRY` using either a *GeometryField* or *RasterField*.

```python
class PipeNetwork(models.Model):
    id = models.AutoField(primary_key=True)
    length_km = models.FloatField(help_text="in kilometers")
    diameter_mm = models.FloatField(help_text="Pipe diameter in
        millimeters", null=True, blank=True)
    diameter_in = models.FloatField(help_text="Pipe diameter in
        inches", null=True, blank=True)
    geom = models.MultiLineStringField(srid=COORDINATE_SYSTEM)
    maitenanceCost_EUR_km = models.FloatField(null=True, help_text
        ="in EUR per kilometer")
    origin = models.ForeignKey(ExtractionWater, on_delete=models.
        DO_NOTHING, help_text="ExtractionWater ID from
        watersupply.ExtractionWater", null=True) # type: ignore
    destination = models.ForeignKey(UsersLocation, on_delete=
        models.DO_NOTHING, help_text="UsersLocation ID from
        common.UsersLocation", null=True) # type: ignore
    last_updated = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"{self.length_km} km"

    def save(self, *args, **kwargs):
        self.length_km = self.geom.length / 1000
        if self.diameter_mm is not None: self.diameter_in = self.
            diameter_mm / 25.4
        if self.diameter_in is not None: self.diameter_mm = self.
            diameter_in * 25.4
        super().save(*args, **kwargs)


    class Meta:
        verbose_name = "Pipe Network"
        verbose_name_plural = "Pipe Networks"

        constraints = [
            models.UniqueConstraint(fields=['origin', 'destination
                '], name='unique_pipe_network')
        ]
```

Algorithm 1: Python Class "PipeNetwork" Model - Code for structuring database tables.

3.2 Import Layer - User Interface

This layer implements a Django form in a React four-page frontend, displayed as shown in Figure 3. The application provides the user with a file upload and attribute mapping interaction. On the first page, it is possible to select the file to be imported into the database; next, a simple file extension validation is performed by the system; upon uploading, the system allows for selecting a planning sector or topic that corresponds to the data to be uploaded (Water supply, urban heat, housing, weather, Builtup) and a target model of destination. Both of which are configured in the Data layer. See also the GitHub Repository https://github.com/ivan-cardenas/GeoImporter.

The uploaded payload is temporarily stored in memory, and the user can then map the uploaded attribute names (from the user data) to the database structure, with required fields highlighted and helper text guiding the user to the units data that should be uploaded. Geometry attributes like object area (for a polygon) or object length (for a line) are automatically calculated on save or update of each record as configured in the Data Layer (Figure 4). Finally, the user will see the number of records (rows) and the status of each record (created, updated, skipped, or error), as shown in Figure 4c.

3.3 Application Layer - Processing data

To transform the uploaded data into the database, several types of parsing are required and performed on the backend for each upload request.

Shapefiles and GeoJSON files are parsed using the GeoPandas Python library, while Raster TIFF files are parsed using the Rasterio library. In both cases, a GEOS (Geometry Engine, Open Source) function is used to transform the coordinate system from the uploaded file to the EPSG code of the desired destination. For vector data, the import algorithm iterates over the rows of the GeoPandas dataframe, resolving foreign keys (for relational models), correcting field types, and performing upsert (update+insert) operations on the database. For raster data, a

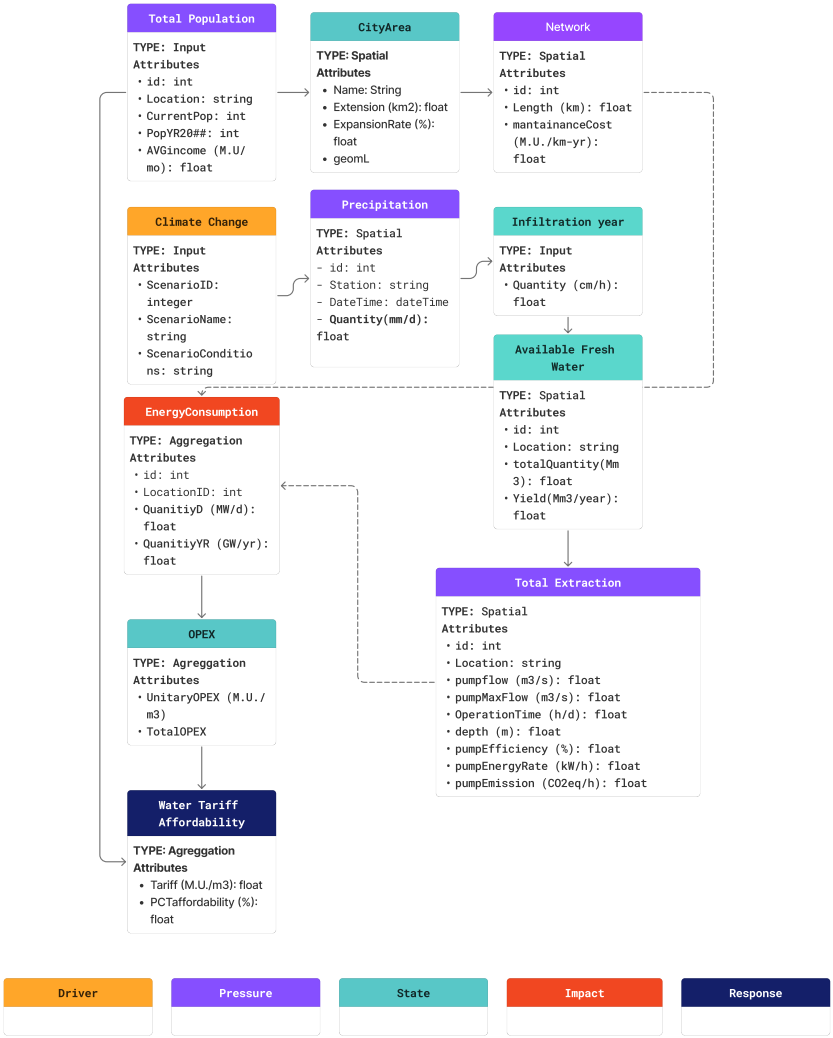


Figure 1: Directed Acyclic Graph - Water Tariff Affordability.
M.U. indicates Monetary Unit (EUR, USD, GPB, COP, etc.)

temporary, reprojected file is stored on disk and then loaded into a RasterField in the database using GDAL.

3.4 Visualization Layer - User Interface

This layer includes a map API that exposes endpoints serving geospatial data to a Mapbox GL JS front-end, with available layers discovered dynamically by introspecting model registries rather than requiring hardcoded data sources. For each registered model, the endpoint determines the geometry type, record count, and attribute fields, and constructs a GeoJSON *FeatureCollection* response directly in the database using PostGIS `STAsGeoJSON` and `STTransform`, reprojecting geometries to WGS84 (EPSG:4326) on the fly for web mapping compatibility. The vector data is, then, retrieved from "`api/ $appLabel>/ <$modelName>/ geojson/`".

Raster data follows a separate serving path. Files are stored on disk as COGs and served as ZXY PNG tiles through TiTiler (Sarago, 2026), a lightweight FastAPI-based tile server that handles on-demand reprojection, colormap application, and value rescaling. The raster makes a call on `"/api/raster/$appLabel/$modelName/tiles/?id=$rasterID"` endpoint that refers to the tiles server on `"{TiTilerBaseURL}/ cog/ tiles/ WebMercatorQuad/{z}/{x}/{y}.png"`. This separation between vector and raster pipelines allows the frontend to consume both data types through a unified layer manifest, while keeping the tile rendering workload outside the Django application server.

4. Discussion and Limitations

The system presented in this paper demonstrates that a working, open-source ingestion pipeline can be constructed from a question-driven data model, without requiring expert SQL knowledge or manual schema definition. By grounding the database schema in a DAG decomposition of the planning question, the system keeps the data structure anchored to its analytical purpose, a property that distinguishes it from general-purpose ETL tools and schema-first database design approaches. The working implementation represents a transferable foundation that can be adapted to new urban contexts by defining a planning question and its measurable dependencies.

The approach taken here sits within a longer tradition of Planning Support Systems (PSS) research, which has long argued that

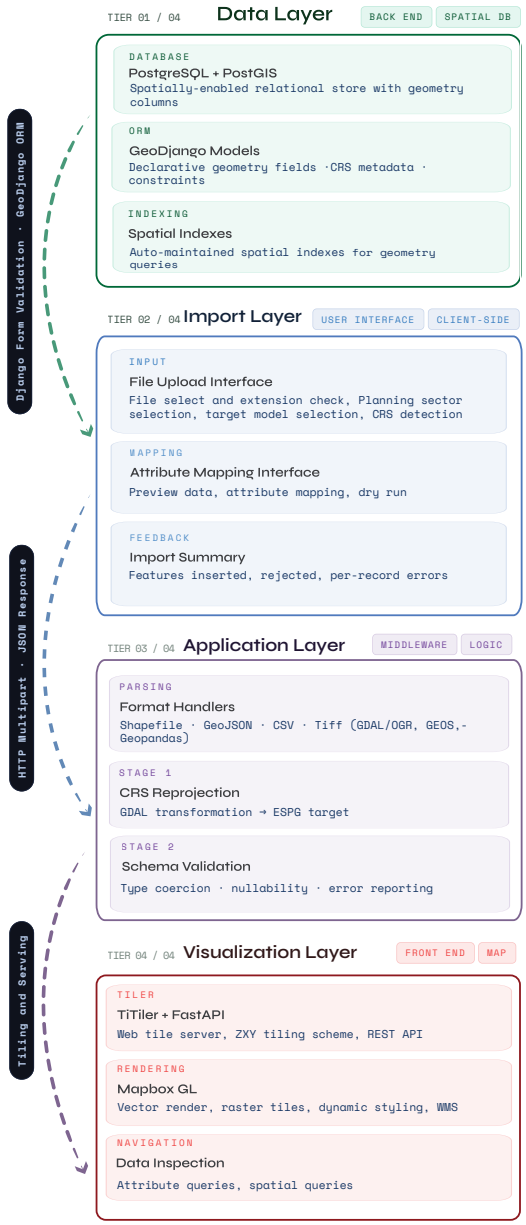


Figure 2: Data Import - System Architecture.

spatial decision support tools must be organized around planning questions rather than data availability (Geertman, 2006; Pelzer et al., 2014). Where earlier PSS work focused primarily on analytical and visualization capabilities, the present contribution addresses an upstream bottleneck that has received less systematic attention: the structured ingestion of heterogeneous geospatial data into a schema that reflects the planning question's logical structure. In this sense, the system can be understood as a building block for PSS development rather than a complete planning support environment.

The proposed decomposition procedure assumes that the planning question can be expressed as a structured dependency among measurable variables, whether deterministic or stochastic. The framework does not accommodate qualitative, contested, or empirically inaccessible variables, such as perceived thermal comfort or informal housing occupancy, as these cannot be assigned the spatial reference, temporal resolution, and units of measurement required for schema generation. Furthermore, the method presupposes sufficient domain knowledge to correctly specify the direction of each dependency; an incorrectly directed edge propagates structural errors throughout the derived schema. These constraints suggest that DAG construction is best treated as a collaborative, iterative process involving domain experts rather than a purely technical procedure.

The model-driven, schema-on-write approach presented here is not the only viable path for handling heterogeneous urban data. Schema-on-read structures, common in data lakes such as Hadoop (Shvachko et al., 2010), move the structure definition until query time. Although this offers greater flexibility for exploring evolving, loosely defined datasets, the structured access, spatial indexing, and validation required for a cross-sectoral impact analysis of a UDT database are not available. Likewise, interactions with libraries such as pyodbc or exploratory prototyping with Pandas, Geopandas, and NumPy offer lower overhead for single-user or one-time ingestion tasks, but do not provide a persistent schema to support recurring, non-expert contributions from different planning sectors into a common database. The ORM-based approach presented here works best in contexts where the schema is stabilized through DAG decomposition of planning questions, where multiple non-expert contributors are expected to populate a database, and where spatial constraints and long-term interoperability across sectors outweigh the flexibility and exploratory capabilities of a schema-on-read or script-based alternative.

Contrasting with the previous work of Fargas Jr. et al. (2024) and Kipkemboi et al. (2023), the proposed framework aims to store data in a self-hosted database and WMS endpoint while avoiding complex SQL scripts for data structuring or ingestion. The system thus becomes independent of other applications, such as GeoServer or OpenDataCube, and the database can be easily connected to traditional GIS software, such as ArcGIS or QGIS, for exploration and analysis.

Several limitations constrain the current implementation and point toward future development priorities. The thematic domains available for data import (water supply, urban heat, housing) are defined at the application layer and require developer intervention to extend. This means that while the ingestion pipeline itself is transferable, adapting the system to a new planning context currently requires a developer to configure the relevant Django applications and model registries. Automating this step, for instance, through a guided DAG editor that generates ORM models directly, would substantially increase accessibility for non-technical users and is identified as the primary direction for future work.

The designed system is limited to ingesting local data and lacks a feature that allows retrieving information directly from open data sources and storing it in the database. Although this functionality requires source-specific structuring to parse data into the database schema, it would be possible to enable reading external APIs and transforming JSON responses into structured spatial objects that conform to the Digital Twin database schema. Likewise, more general data can be obtained from sources such as OpenStreetMap, Google Earth Engine, AWS, or direct connections to satellite imagery providers' APIs such as SentinelHub, USGS, or Planet (the last one requires a subscription and authentication).

There is also a limitation on the size of the files to be ingested. Very large GeoTIFFs or long datasets can cause memory issues because the files are loaded into memory in their entirety

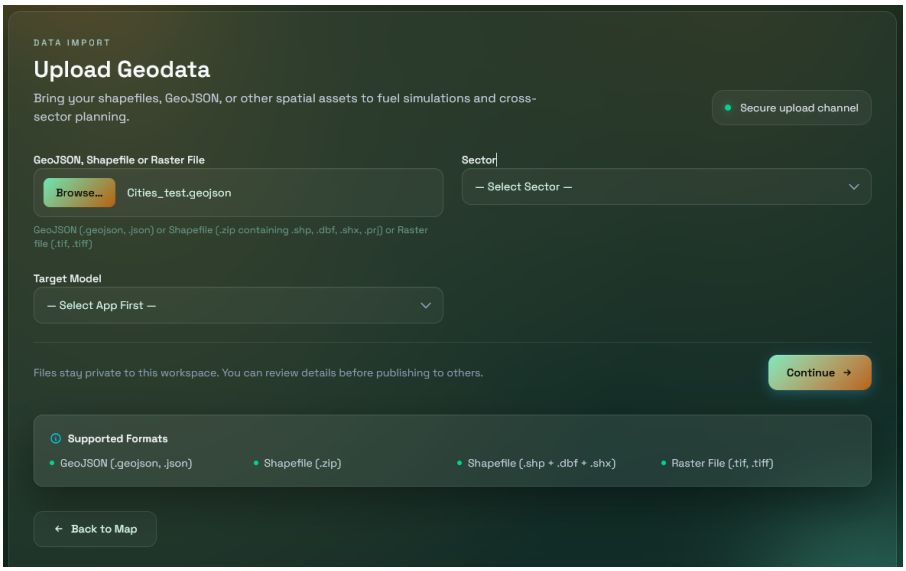
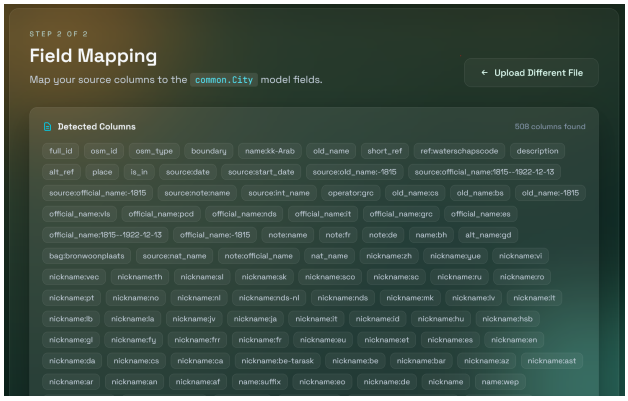
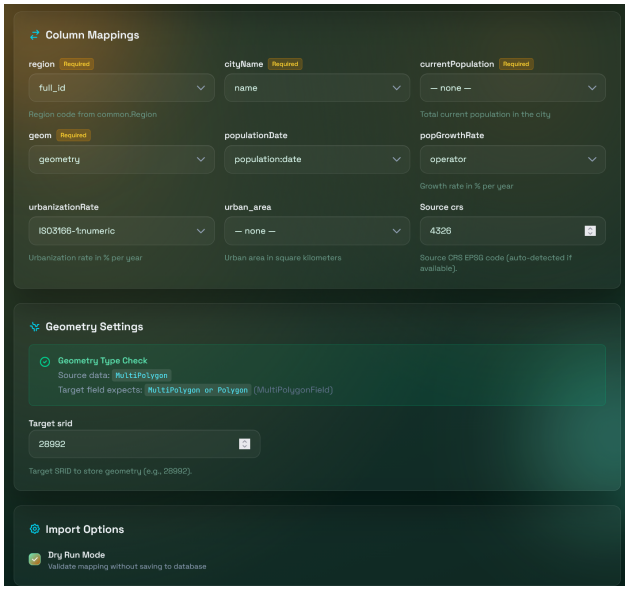


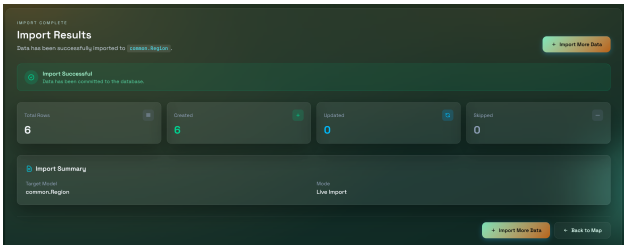
Figure 3: Upload Geodata form Frontend.



(a) Detected columns from source data



(b) Column mapping and Geometry reprojection settings



(c) Import Results

Figure 4: Data Mapping Frontend - Example using OSM Administrative boundaries.

before being imported into the database. Likewise, files need to be imported one by one for each model, as no batch import is available. Similarly, the user must identify which source column corresponds to each model field, as fuzzy matching is not supported. Geometry issues, such as out-of-bounds and duplicate records, are not flagged, and no topology validation is implemented before insertion into the PostGIS database.

Also, the system still requires configuration to support the ingestion of real-time sensor data from SensorThings (Liang et al., 2021) or NGSI-LD (ETSI ISG CIM, 2025) protocols, where, for instance, PostGIS *GEOMETRY* attributes can map to ”GeoProperty” or ”Locations”, the *TIMESTAMP* can map to ”observedAt” or ”Observations” timeseries. In this way, it would be possible to evaluate queries on boundary-defined polygons to retrieve sensor changes in that area and perform spatial math operations and derivatives using SQL calculations that would have slow performance in a NoSQL context.

5. Conclusions

This paper presented a manual methodology and a web-based geospatial data importer to systematically translate spatial planning questions into configured spatial databases for Urban Digital Twin development. The main contribution we present here is the front-end importer for a unified spatial database. The approach proceeds in two stages: first, formalizing the planning question as a Directed Acyclic Graph whose nodes define measurable variables and whose edges encode functional dependencies; second, deriving a relational database schema from that graph and providing a schema-aware ingestion interface for heterogeneous geospatial inputs.

The system demonstrates that Django’s ORM can be used to derive import specifications directly from the data model, thereby

reducing the need for hard-coded schema definitions and keeping the application and database layers synchronized. By serving imported data through a PostGIS-backed GeoJSON API and a TiTiler raster tile server, the system further connects ingestion directly to visualization, enabling a continuous workflow from raw file upload to interactive map display in MapboxGL JS.

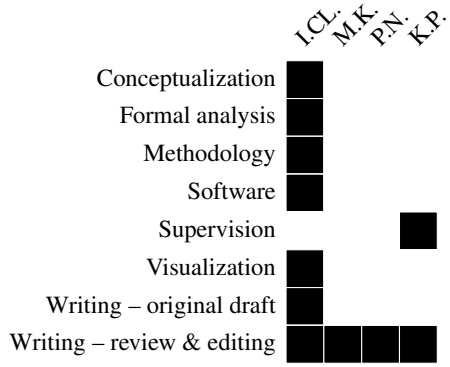
Future work will focus on performing cross-sectoral calculations based on spatial features to answer the planning questions at hand. This will be done in a unified scenario-testing environment as an UDT in which the configured spatial database can be queried to produce direct, evidence-based answers to the planning questions that motivated its construction. Additionally, future extension would integrate a controlled natural-language interface that translates user queries into model-specific requests aligned with the attributes, units, and predefined data sources represented in the database schema. With this, it would also be possible to issue a REST request to retrieve data from a predefined list of external sources (Google Earth Engine, Statistics department, web feature services, etc.), and then import it into the Digital Twin using the specific database schema defined in the DAG.

As UDT platforms mature, a robust, flexible data ingestion infrastructure will remain a prerequisite for their effective deployment in real-world urban planning contexts. Transforming a planning question into a structured database through a DAG-based decomposition process is a key step toward creating sense-giving planning tools that remain anchored to the questions they are designed to answer, rather than becoming a large repository of unstructured data.

6. Acknowledgments

This research was supported by the Dutch Sectorplan Bèta II programme in Earth and Environmental Sciences (AMW). For more information, see: https://www.sectorplan-betatechniek.nl

CRediT statement



References

Aime, A.; Fabiani, A.; Hards, B.; Bühner, N.; Romagnoli, D.; Tajariol, E.; Miño, F.; Roldan, G.; Turton, I.; Hughes, J.; Garnett, J.; Fix, J.; Rahkonen, J.; Prins, M.; Oliveira, N.; Rushforth, P.; Parma, A.; Ikeoka, S.; Giannecchini, S.; Smith, K.; Barsballe, T.; GeoSolutions SRL; GeoCat BV & Open Source Geospatial Foundation (2026); GeoServer; https://geoserver.org (09-March-2026).

Alegre, H.; Jr, E. C.; Duarte, P.; Merkel, W.; Baptista, J. M.; Cubillo, F.; Hirner, W. & PArena, R. (2017); Indicadores de desempeño para servicios de abastecimiento de agua [in Spanish]; http://dx.doi.org/https://doi.org/10.2166/9788490486641.

Allan, M.; Rajabifard, A.; Chen, B. & Foliente, G. (2025); Utilising urban digital twins for sustainable urban regeneration: Melbourne's Greenline planning and assessment framework; Sustainable Cities and Society; vol. 131: p. 106,661; ISSN 22106707; http://dx.doi.org/10.1016/j.scs.2025.106661 (24-February-2026).

Alvi, M.; Dutta, H.; Minerva, R.; Crespi, N.; Raza, S. M. & Herath, M. (2025); Global perspectives on digital twin smart cities: Innovations, challenges, and pathways to a sustainable urban future; Sustainable Cities and Society; vol. 126: p. 106,356; ISSN 22106707; http://dx.doi.org/10.1016/j.scs.2025.106356 (24-February-2026).

Caprari, G.; Castelli, G.; Montuori, M.; Camardelli, M. & Malvezzi, R. (2022); Digital Twin for Urban Planning in the Green Deal Era: A State of the Art and Future Perspectives; Sustainability; vol. 14 (10): p. 6263; ISSN 2071-1050; http://dx.doi.org/10.3390/su14106263 (06-January-2024).

Chapagain, K.; Aboelnga, H.; Babel, M.; Ribbe, L.; Shinde, V.; Sharma, D. & Dang, N. (2022); Urban water security: A comparative assessment and policy analysis of five cities in diverse developing countries of Asia; Environmental Development; vol. 43; http://dx.doi.org/10.1016/j.envdev.2022.100713.

Dembski, F.; Wössner, U.; Letzgus, M.; Ruddat, M. & Yamu, C. (2020); Urban Digital Twins for Smart Cities and Citizens: The Case Study of Herrenberg, Germany; Sustainability; vol. 12 (6): p. 2307; ISSN 2071-1050; http://dx.doi.org/10.3390/su12062307 (19-March-2024).

Django Software Foundation (2025); Django (version 6.0); Django documentation | Django documentation; https://www.djangoproject.com/ (23-February-2026).

ETSI ISG CIM (2025); Context Information Management (CIM); NGSI-LD API; Group Specification ETSI GS CIM 009 V1.9.1; European Telecommunications Standards Institute (ETSI); https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.09.01_60/gs_cim009v010901p.pdf.

European Environment Agency (1999); Environmental indicators:Typology and overview; Tech. rep.; European Environment Agency; https://www.eea.europa.eu/en/analysis/publications/tec25 (28-January-2025).

Fargas Jr., D. C.; De La Cruz, R. M. & Blanco, A. C. (2024); Space+ Data Dashboard: Empowering Institutions and Citizens with Seamless Space Data Access; The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences; vol. XLVIII-4/W8-2023: pp. 219–225; ISSN 2194-9034; http://dx.doi.org/10.5194/isprs-archives-XLVIII-4-W8-2023-219-2024 (24-February-2026).

Geertman, S. (2006); Potentials for Planning Support: A Planning-Conceptual Approach; Environment and Planning B: Planning and Design; vol. 33 (6): pp. 863–880; ISSN 0265-8135; http://dx.doi.org/10.1068/b31129 (30-January-2024).

Grieves, M. (2015); Digital Twin: Manufacturing Excellence through Virtual Factory Replication.

Horak, J.; Jurikovska Orlikova, L.; Umlauf, M.; Ježek, J.; Hanzlová, M. & Zlatanova, S. (2011); HUMBOLDT alignment editor; in *GIS Ostrava 2011*; https://www.researchgate.net/publication/254908528_HUMBOLDT_alignment_editor.

Jeddoub, I.; Nys, G.-A.; Hajji, R. & Billen, R. (2023); Digital Twins for cities: Analyzing the gap between concepts and current implementations with a specific focus on data integration; International Journal of Applied Earth Observation and Geoinformation; vol. 122: p. 103,440; ISSN 1569-8432; http://dx.doi.org/10.1016/j.jag.2023.103440 (22-January-2024).

Jeddoub, I.; Nys, G.-A.; Hajji, R. & Billen, R. (2025); Data integration across urban digital twin lifecycle: a comprehensive review of current initiatives; Annals of GIS; vol. 31 (3): pp. 367–386; ISSN 1947-5683; http://dx.doi.org/10.1080/19475683.2024.2416135 (09-March-2026).

Ketzler, B.; Naserentin, V.; Latino, F.; Zangelidis, C.; Thuvander, L. & Logg, A. (2020); Digital Twins for Cities: A State of the Art Review; Built Environment; vol. 46 (4): pp. 547–573; http://dx.doi.org/10.2148/benv.46.4.547.

Kipkemboi, W.; Kuria, B. T.; Kuria, D. N.; Sichangi, A. W.; Mundia, C. N.; Wanjala, J. A.; Muthee, S. W.; Goebel, M. & Rienow, A. (2023); Development of a Web-GIS Platform for Environmental Monitoring and Conservation of the Muringato Catchment in Kenya; Journal of Geovisualization and Spatial Analysis; vol. 7 (1): p. 13; ISSN 2509-8810, 2509-8829; http://dx.doi.org/10.1007/s41651-023-00143-3 (24-February-2026).

Kritzinger, W.; Karner, M.; Traar, G.; Henjes, J. & Sihn, W. (2018); Digital Twin in manufacturing: A categorical literature review and classification; IFAC-PapersOnLine; vol. 51 (11): pp. 1016–1022; ISSN 2405-8963; http://dx.doi.org/10.1016/j.ifacol.2018.08.474 (17-May-2024).

Lei, B.; Janssen, P.; Stoter, J. & Biljecki, F. (2023); Challenges of urban digital twins: A systematic review and a Delphi expert survey; Automation in Construction; vol. 147: p. 104,716; ISSN 0926-5805; http://dx.doi.org/10.1016/j.autcon.2022.105866 (04-January-2024).

Liang, S.; Khalafbeigi, T. & van der Schaaf, H. (2021); OGC SensorThings API Part 1: Sensing Version 1.1; OGC Implementation Standard 18-088; Open Geospatial Consortium (OGC); https://ogc.org.

Mandal, D.; Zou, L.; Wadhwa, A.; Wilkho, R. S.; Cai, Z.; Zhou, B.; Ye, X.; Newman, G.; Gharaibeh, N. & Güneralp, B. (2026); FlowsDT: A geospatial digital twin for navigating urban flood dynamics; Computers, Environment and Urban Systems; vol. 126: p. 102,414; ISSN 01989715; http://dx.doi.org/10.1016/j.compenvurbsys.2026.102414 (03-March-2026).

Patel, U. R.; Ghaffarianhoseini, A.; GhaffarianHoseini, A. & Burgess, A. (2026); Towards a well-being-oriented framework for urban digital twins; Cities; vol. 169: p. 106,579; ISSN 02642751; http://dx.doi.org/10.1016/j.cities.2025.106579 (24-February-2026).

Pearl, J. (2009); Causal inference in statistics: An overview; Statistics Surveys; vol. 3 (none); ISSN 1935-7516; http://dx.doi.org/10.1214/09-SS057 (02-March-2026).

Pelzer, P.; Geertman, S.; van der Heijden, R. & Rouwette, E. (2014); The added value of planning support systems: A practitioner's perspective; Computers, Environment and Urban Systems; vol. 48: pp. 16–27; ISSN 01989715; http://dx.doi.org/10.1016/j.compenvurbsys.2014.05.002.

Potts, R. & Webb, B. (2023); Digital planning practices: benchmarking planners' use of information and communication technologies (ICTs); Planning Practice & Research; vol. 38 (4): pp. 520–540; ISSN 0269-7459; http://dx.doi.org/10.1080/02697459.2023.2216492 (24-February-2026).

Quek, H. Y.; Sielker, F.; Akroyd, J.; Bhave, A. N.; Richthofen, A. v.; Herthogs, P.; Yamu, C. v. d. L.; Wan, L.; Nochta, T.; Burgess, G.; Lim, M. Q.; Mosbach, S. & Kraft, M. (2023); The conundrum in smart city governance: Interoperability and compatibility in an ever-growing ecosystem of digital twins; Data & Policy; vol. 5: p. e6; ISSN 2632-3249; http://dx.doi.org/10.1017/dap.2023.1 (24-February-2026).

Sarago, V. (2026); TiTiler (version 1.2.0); https://developmentseed.org/titiler/ (23-February-2026).

Shafto, M.; Conroy, M.; Doyle, R.; Glaessgen, E.; Kemp, C.; LeMoigne, J. & Wang, L. (2010); DRAFT Modeling, Simulation, Information, Technology & Processing Roadmap; https://www.emacromall.com/reference/NASA-Modeling-Simulation-IT-Processing-Roadmap.pdf.

Shvachko, K.; Kuang, H.; Radia, S. & Chansler, R. (2010); The hadoop distributed file system; in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*; pp. 1–10; http://dx.doi.org/10.1109/MSST.2010.5496972.

Wang, L.; Zhang, X. & He, W. (2026); The construction and optimization of resilient community living sphere driven by digital twins; Scientific Reports; vol. 16 (1): p. 4268; ISSN 2045-2322; http://dx.doi.org/10.1038/s41598-025-34455-9 (24-February-2026).

wetransform GmbH & halestudio contributors (2024); hale studio (HUMBOLDT Alignment Editor); https://www.wetransform.to/downloads/.

White, C. T.; Petrasova, A.; Petras, V.; Tateosian, L. G.; Vukomanovic, J.; Mitasova, H. & Meentemeyer, R. K. (2023); An open-source platform for geospatial participatory modeling in the cloud; Environmental Modelling & Software; vol. 167: p. 105,767; ISSN 13648152; http://dx.doi.org/10.1016/j.envsoft.2023.105767 (09-March-2026).

White, G.; Zink, A.; Codecá, L. & Clarke, S. (2021); A digital twin smart city for citizen feedback; Cities; vol. 110: p. 103,064; ISSN 0264-2751; http://dx.doi.org/10.1016/j.cities.2020.103064 (17-September-2025).

Wysocki, O.; Schwab, B.; Biswanath, M. K.; Greza, M.; Zhang, Q.; Zhu, J.; Froech, T.; Heeramaglore, M.; Hijazi, I.; Kanna, K.; Pechinger, M.; Chen, Z.; Sun, Y.; Segura, A. R.; Xu, Z.; AbdelGafar, O.; Mehranfar, M.; Yeshwanth, C.; Liu, Y.-C.; Yazdi, H.; Wang, J.; Auer, S.; Anders, K.; Bogenberger, K.; Borrmann, A.; Dai, A.; Hoegner, L.; Holst, C.; Kolbe, T. H.; Ludwig, F.; Nießner, M.; Petzold, F.; Zhu, X. X. & Jutzi, B. (2026); TUM2TWIN: Introducing the large-scale multimodal urban digital twin benchmark dataset; ISPRS Journal of Photogrammetry and Remote Sensing; vol. 232: pp. 810–830; ISSN 09242716; http://dx.doi.org/10.1016/j.isprsjprs.2025.12.013 (24-February-2026).