Project Data Dictionary

Federica Gaspari

Table of contents

1. Welcome to the Data Dictionary Hub 3
1.1 Our Objectives 3
1.2 How to Navigate 3
1.3 How to Contribute 3
2. Schema overview 4
2.1 1. Enterprises and ownership 4
2.2 2. Components and their sub-types 4
2.3 3. Interactions between entities 4
3. Data Dictionary 6
3.1 Entities and relationships 6
4. Contribution Guidelines 76
4.1 The Workflow 76
4.2 How to fill out the Excel template 76
4.3 Submitting Your Changes 76
5. Graphic User Interface terminology 77
5.1 Backend to web-app correspondence 77

1. Welcome to the Data Dictionary Hub

Welcome to the central documentation platform for our project's database.

This platform is designed to make our data structure transparent, accessible, and easy to understand for everyone involved in the project—whether you are a developer, a data analyst, or a domain expert.

1.1 Our Objectives

The main goals of this data dictionary are:

- **Single Source of Truth:** Provide a reliable and standardized reference for all database entities, their attributes, and relationships.
- **Accessibility:** Offer a simple, fast, and easily shareable web interface. No complex software installations or database credentials are required to understand our data.
- **Clarity & Context:** Go beyond just "data types". Here you will find business definitions, constraints, example values, and specific rules (e.g., geographical data handling) for every single column.
- **Collaborative Maintenance:** Empower non-technical team members to keep the documentation alive and updated through a simplified workflow.

1.2 How to Navigate

Finding the information you need is quick and straightforward:

1. **Search Bar:** Use the powerful search bar at the top of the page. It instantly looks through table names, column definitions, and comments.
2. **Interactive Schema:** Start with the Schema Overview (ER) to get a bird's-eye view of how different tables relate to each other.
3. **Table Details:** Browse the sidebar navigation menu to dive deep into specific domains and tables. Each page provides an exhaustive breakdown of the selected table.

1.3 How to Contribute

Documentation is only useful if it stays up to date. We built this platform so that **anyone can contribute, no coding required**.

!!! info "Updating the Dictionary via Excel"

```
You don't need to know Markdown or Git to help out! We manage updates via a simple **Excel template**.

If you need to define a new table or update an existing description, you just need to fill out a spreadsheet containing columns like *Data type, Definition, Example value, etc.*

The technical team will automatically convert your Excel sheet into these web pages. Please read the [Contribution Guidelines](guidelines.md) for more details.
```

If you find any discrepancies or have questions about specific data entities, please reach out to the database administration team.

2. Schema overview

The `public` schema is organised around three groups of tables:

1. **Enterprises**: the owned, located units that can be damaged (buildings, farms, businesses, network systems).
2. **Components**: the physical or economic elements that belong to an enterprise (crops, machinery, staff, road segments, and so on).
3. **Assessment**: events, surveys and the damage records they produce.

Enterprises and components both use **table-per-subtype inheritance**: a sub-type table shares the primary key of its parent, so a row is joined to its parent with `child.id = parent.id`.

The schema is shown in three diagrams to keep each one readable.

2.1 1. Enterprises and ownership

```
erDiagram
    OWNER ||--o{ ENTERPRISE : owns
    ENTERPRISE ||--o| BUILDING : "is a"
    ENTERPRISE ||--o| BUSINESS : "is a"
    ENTERPRISE ||--o| FARM : "is a"
    ENTERPRISE ||--o| NETWORK_SYSTEM : "is a"
    NETWORK_SYSTEM ||--o| ELECTRICAL_SYSTEM : "is a"
    NETWORK_SYSTEM ||--o| TLC_SYSTEM : "is a"
    NETWORK_SYSTEM ||--o| TRANSPORT_SYSTEM : "is a"
    ENTERPRISE ||--o{ COMPONENT : contains
    ENTERPRISE ||--o{ SURVEY : undergoes
```

2.2 2. Components and their sub-types

```
erDiagram
    ENTERPRISE ||--o{ COMPONENT : contains
    COMPONENT ||--o{ DAMAGE : suffers

    COMPONENT ||--o| ADDED_BLOCK : "is a"
    COMPONENT ||--o| ARTISTIC_GOOD : "is a"
    COMPONENT ||--o| BUILDING_COMPONENT : "is a"
    COMPONENT ||--o| BUILDING_FUNCTIONALITY : "is a"
    COMPONENT ||--o| BUSINESS_OPERATIONS : "is a"
    COMPONENT ||--o| LIVESTOCK : "is a"
    COMPONENT ||--o| NETWORK_SERVICE : "is a"
    COMPONENT ||--o| PRODUCT : "is a"
    COMPONENT ||--o| PROVISION : "is a"
    COMPONENT ||--o| STAFF : "is a"
    COMPONENT ||--o| ELECTRICAL_NETWORK_COMPONENT : "is a"
    COMPONENT ||--o| TLC_NETWORK_COMPONENT : "is a"
    COMPONENT ||--o| TRANSPORT_NETWORK_COMPONENT : "is a"

    COMPONENT ||--o| AGRICULTURE_PRODUCT : "is a"
    AGRICULTURE_PRODUCT ||--o| CROP : "is a"
    AGRICULTURE_PRODUCT ||--o| STORED : "is a"

    COMPONENT ||--o| FIXED_ASSET : "is a"
    FIXED_ASSET ||--o| INFRASTRUCTURE : "is a"
    FIXED_ASSET ||--o| LAND : "is a"
    FIXED_ASSET ||--o| MACHINERY : "is a"
    FIXED_ASSET ||--o| MATERIAL : "is a"
```

2.3 3. Interactions between entities

```
erDiagram
    EVENT ||--o{ SURVEY : triggers
    ENTERPRISE ||--o{ SURVEY : undergoes
    SURVEY ||--o{ DAMAGE : reports
    COMPONENT ||--o{ DAMAGE : suffers
    ENTERPRISE ||--o{ COMPONENT : contains
    EVENT ||--o{ EVENT : linked
    DAMAGE ||--o{ ECONOMIC_LOSS : has
    OWNER ||--o{ ENTERPRISE_OWNER : owns
    ENTERPRISE ||--o{ ENTERPRISE_OWNER : is_owned
```

2.3.1 Relationship Legend

- `||--o{` : **One-to-Many**. *Example: An enterprise contains many components.*
- `||--||` : **One-to-One**. *Example: The Building record is a direct extension of an Enterprise record.*
- `}o--||` : **Many-to-One**. *Example: Many enterprises can belong to a single owner.*

3. Data Dictionary

3.1 Entities and relationships

3.1.1 Enterprise and subclasses

Table: `enterprise`

Description (EN): This is the core parent entity (superclass) of the database. It stores the universal, high-level attributes, such as identification, geographical location and administrative details, for all major physical or business assets. Specific types of enterprises (like buildings, businesses, farms or network systems) store their specialized data in separate sub-tables that share the same ID. Ownership is not stored here: it is recorded in the bridge table `enterprise_owner`.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the enterprise	`3`	PK	No
`type`	VARCHAR(255)	Discriminator column defining the specific type of enterprise	`BUILDING`	NOT NULL	No
`name`	VARCHAR(255)	Official name or denomination of the enterprise	`Main Headquarters`		No
`management_type`	VARCHAR(255)	Type of management or business administration	`PUBLIC`		No
`vat`	VARCHAR(255)	VAT number of the enterprise	`IT00000000000`		No
`description`	VARCHAR(255)	Free-text description of the enterprise	`Regional hospital`		No
`country`	VARCHAR(255)	Country where the enterprise is located	`Italy`		No
`state`	VARCHAR(255)	State or region where the enterprise is located	`Lazio`		No
`county`	VARCHAR(255)	County or province where the enterprise is located	`Rome`		No
`town`	VARCHAR(255)	Municipality where the enterprise is located	`Rome`		No
`hamlet`	VARCHAR(255)	Hamlet or locality where the enterprise is located	`Trastevere`		No
`road`	VARCHAR(255)	Street where the enterprise is located	`Via Appia`		No
`house_number`	VARCHAR(255)	House number of the enterprise	`10`		No
`postcode`	VARCHAR(255)	Postal code of the enterprise	`00100`		No
`geometry`	GEOMETRY	Spatial representation of the enterprise	`POINT(12.49 41.89)`		Yes

Column	Data type	Definition	Example value	Constraint?	Geometry?
`coordinates_inferred`	BOOLEAN	Whether the coordinates of the enterprise were inferred rather than directly measured	[...]		No

RELATIONSHIPS

Because `enterprise` is the central hub of the schema, it has extensive relationships:

- **Has Sub-types (Inheritance):** The following tables inherit from `enterprise` and share its `id`:
  - `building`
  - `business`
  - `farm`
  - `network_system`
- **Is owned by:** `owner`. An enterprise can have several owners, and an owner can hold several enterprises. This many-to-many relationship is mediated by the `enterprise_owner` bridge table.
- **Has many:** `component` (An enterprise is composed of multiple sub-components like machinery, crops, or electrical networks)
- **Has many:** `survey` (An enterprise can undergo multiple damage or assessment surveys over time)

EXAMPLE QUERY

Retrieve a list of all enterprises located in a specific town, displaying their name, their specific classification (type), and their coordinates:

```
SELECT
    id,
    name,
    type,
    ST_AsText(geometry) as coordinates
FROM
    enterprise
WHERE
    town = 'Rome'
ORDER BY
    type ASC;
```

Retrieve the owners of each enterprise through the bridge table:

```
SELECT
    e.id,
    e.name,
    o.full_name
FROM
    enterprise e
JOIN
    enterprise_owner eo ON eo.enterprise_id = e.id
JOIN
    owner o ON o.id = eo.owner_id;
```

Table: `building`

Description (EN): This table represents buildings. In the database's inheritance architecture, it acts as a direct sub-type of the `enterprise` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `enterprise` record.

The structural, economic and physical characteristics of a building (construction year, materials, number of floors, cost, and so on) are stored in the `building_component` table.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Building unique identifier	`3`	PK, FK	No	Primar also ac Foreign referen `enterp`

RELATIONSHIPS

- **Inherits from (Sub-type of):** `enterprise`. The `building` table is a specialized extension of the `enterprise` table.
- **Contains:** While not linked directly via a `building_id`, a building owns its physical parts through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `building_component`
  - `added_block`
  - `building_functionality`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or VAT of a building, you must join this table with the `enterprise` record that shares the exact same `id`. The structural and economic characteristics are in `building_component`, which is linked to the building through `component.enterprise_id`.

EXAMPLE QUERY

Retrieve the characteristics of each building by joining its enterprise record with its building components:

```
SELECT
    e.name,
    e.town,
    bc.n_floors,
    bc.construction_cost,
    bc.economic_value
FROM
    building b
JOIN
    enterprise e ON b.id = e.id
JOIN
    component c ON c.enterprise_id = e.id
JOIN
    building_component bc ON bc.id = c.id
WHERE
    e.type = 'BUILDING';
```

Table: `business`

Description (EN): This table represents business enterprises. In the database's inheritance architecture, it acts as a direct sub-type of the `enterprise` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `enterprise` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the business record	`8`	PK, FK	No	Primar also ac Foreign referen `enterp`
`share_capital`	BIGINT	Total monetary value of funds of the company	[...]	-	No	[...]
`nace_section`	VARCHAR	Top-level industry classification of the business's primary economic activity, according to the NACE standard	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `enterprise`. The `business` table is a specialized extension of the `enterprise` table.
- **Contains:** While not linked directly via a `business_id`, a network system owns infrastructural assets through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `business_operations`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a network system, you must join this table with the `enterprise` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all business records, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    b.id,
    e.name,
    e.town,
    ST_AsText(e.geometry) as coordinates
FROM
    business b
JOIN
    enterprise e ON b.id = e.id
WHERE
    e.type = 'BUSINESS';
```

Table: `farm`

Description (EN): This table represents agricultural enterprises. In the database's inheritance architecture, it acts as a direct sub-type of the `enterprise` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `enterprise` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the farm	`42`	PK, FK	No	Primar also ac Foreign referen `enterp`

RELATIONSHIPS

- **Inherits from (Sub-type of):** `enterprise`. The `farm` table is a specialized extension of the `enterprise` table.
- **Contains:** While not linked directly via a `farm_id`, a farm owns agricultural assets through the `enterprise_id` column in the `component` table. These components often include specialized agricultural sub-types such as:
  - `agriculture_product` and its sub-types `crop` and `stored`
  - `livestock`
  - `fixed_asset` and its sub-types (e.g. `machinery` such as tractors)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a farm, you must join this table with the `enterprise` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all farms, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    f.id,
    e.name,
    e.town,
    ST_AsText(e.geometry) as coordinates
FROM
    farm f
JOIN
    enterprise e ON f.id = e.id
WHERE
    e.type = 'FARM';
```

Network Systems and sub-types

TABLE: `network_system`

Description (EN): This table represents infrastructural network systems (e.g., transport, electrical, or telecommunication grids). In the database's inheritance architecture, it acts as a direct sub-type of the `enterprise` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `enterprise` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the network system	`8`	PK, FK	No	Primar also ac Foreign referen `enterp`

Relationships

- **Inherits from (Sub-type of):** `enterprise`. The `network_system` table is a specialized extension of the `enterprise` table.
- **Has Sub-types (Inheritance):** The following tables inherit from `network_system` (and, through it, from `enterprise`) and share its `id`:
  - `electrical_system`
  - `tlc_system`
  - `transport_system`
- **Contains:** While not linked directly via a `network_system_id`, a network system owns infrastructural assets through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `transport_network_component`
  - `electrical_network_component`
  - `tlc_network_component`
  - `network_service`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a network system, you must join this table with the `enterprise` record that shares the exact same `id`.

Example Query

Retrieve a list of all network systems, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    n.id,
    e.name,
    e.town,
    e.type,
    ST_AsText(e.geometry) as coordinates
FROM
    network_system n
JOIN
    enterprise e ON n.id = e.id;
```

TABLE: `electrical_system`

Description (EN): This table represents electrical network systems. In the database's inheritance architecture, it acts as a direct sub-type of the `network_system` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `network_system`, which on turn is a child of a `enterprise` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the electrical network system	`8`	PK, FK	No	Primar also ac Foreign referen `networ` and `en`

Relationships

- **Inherits from (Sub-type of):** `network_system`. The `electrical_system` table is a specialized extension of the `network_system` table.
- **Contains:** While not linked directly via an `electrical_system_id`, an electrical system owns infrastructural assets through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `electrical_network_component`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a network system, you must join this table with the `enterprise` record that shares the exact same `id`.

Example Query

Retrieve a list of all electrical systems, including their names, towns, and spatial coordinates, by joining the table with its parent `network_system` - `enterprise` entity:

```sql SELECT es.id, e.name, e.town, ST_AsText(e.geometry) as coordinates FROM electrical_system es JOIN enterprise e ON es.id = e.id WHERE e.type = 'ELECTRICAL_SYSTEM';
```

TABLE: `transport_system`

Description (EN): This table represents transport network systems. In the database's inheritance architecture, it acts as a direct sub-type of the `network_system` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `network_system`, which on turn is a child of a `enterprise` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the transport network system	`8`	PK, FK	No	Primar also ac Foreign referen `networ` and `en`

Relationships

- **Inherits from (Sub-type of):** `network_system`. The `electrical_system` table is a specialized extension of the `network_system` table.
- **Contains:** While not linked directly via a `transport_system_id`, a transport system owns infrastructural assets through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `transport_network_component`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a network system, you must join this table with the `enterprise` record that shares the exact same `id`.

Example Query

Retrieve a list of all electrical systems, including their names, towns, and spatial coordinates, by joining the table with its parent `network_system` - `enterprise` entity:

```sql SELECT tr.id, e.name, e.town, ST_AsText(e.geometry) as coordinates FROM transport_system tr JOIN enterprise e ON tr.id = e.id WHERE e.type = 'TRANSPORT';
```

TABLE: `tlc_system`

Description (EN): This table represents telecommunications network systems. In the database's inheritance architecture, it acts as a direct sub-type of the `network_system` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `network_system`, which on turn is a child of a `enterprise` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the telecommunication network system	`8`	PK, FK	No	Prima also Forei refer `netw` and

Relationships

- **Inherits from (Sub-type of):** `network_system`. The `tlc_system` table is a specialized extension of the `network_system` table.
- **Contains:** While not linked directly via a `tlc_system_id`, a telecommunication system owns infrastructural assets through the `enterprise_id` column in the `component` table. These components often include specialized sub-types such as:
  - `tlc_network_component`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a network system, you must join this table with the `enterprise` record that shares the exact same `id`.

Example Query

Retrieve a list of all electrical systems, including their names, towns, and spatial coordinates, by joining the table with its parent `network_system` - `enterprise` entity:

```sql SELECT tlc.id, e.name, e.town, ST_AsText(e.geometry) as coordinates FROM tlc_system tlc JOIN enterprise e ON tlc.id = e.id WHERE e.type = 'TLC_SYSTEM';
```

3.1.2 Table: `owner`

Description (EN): This entity stores the universal, high-level attributes—such as identification and contacts for owner records.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Com
`id`	BIGINT	Unique identifier for the owner	`3`	PK	No	-
`email`	VARCHAR	Email address of the owner	`example@mail.com`	-	No	-
`fiscal_code`	VARCHAR	Fiscal code of the owner	`AB0123CDEF`	-	No	Char and can v coun coun
`full_name`	VARCHAR	Full name of the owner	`John Doe`	-	No	-
`phone`	VARCHAR	Phone number and international prefix of the owner	`+390123456789`	-	No	-

Relationships

- **Owns:** `enterprise` (An owner owns an enterprirse). This many-to-many relationship is mediated with the `enterprise_owner` bridge table.

Example Query

Retrieve a list of all owner, displaying their name and their specific contacts:

```
SELECT *
FROM
    owner o
ORDER BY
    o.full_name ASC;
```

3.1.3 Components and subclasses

Table: `component`

Description (EN): This table stores the universal, high-level attributes, such as identification and geographical location, for all major physical or business components. Specific types of components (like added blocks, artistic goods, or other) store their specialized data in separate sub-tables that share the same ID.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the component	`3`	PK	No
`component_kind`	VARCHAR	Discriminator column identifying the specific sub-type of the component	`BUILDING_COMPONENT`	NOT NULL	No
`enterprise_id`	BIGINT	Identifier of the enterprise the component belongs to	`42`	FK	No
`insured`	BOOLEAN	Whether the component is covered by insurance	`True`	-	No
`name`	VARCHAR	Name or denomination of the component	`Warehouse A`	-	No
`country`	VARCHAR	Country where the component is located	`Italy`	-	No
`state`	VARCHAR	State or region where the component is located	`Lazio`	-	No
`county`	VARCHAR	County or province where the component is located	`Rome`	-	No
`town`	VARCHAR	Municipality where the component is located	`Rome`	-	No
`hamlet`	VARCHAR	Hamlet or locality where the component is located	`Trastevere`	-	No
`road`	VARCHAR	Street where the component is located	`Via Appia`	-	No
`house_number`	VARCHAR	House number of the component	`10`	-	No
`postcode`	VARCHAR	Postal code of the component	`00100`	-	No
`details`	VARCHAR	Additional address details		-	No

Column	Data type	Definition	Example value	Constraint?	Geometry?
		of the component	`Building B, second floor`		
`geometry`	GEOMETRY	Spatial representation of the component entity	[...]	-	Yes
`coordinates_inferred`	BOOLEAN	Whether the coordinates of the component were inferred rather than directly measured	[...]	-	No
`cultural_heritage`	BOOLEAN	Whether the component is a cultural heritage asset	[...]	-	No

RELATIONSHIPS

Because `component` is one of the core entities of the schema, it has extensive relationships:

- **Has Sub-types (Inheritance):** The following tables inherit from `component` and share its `id`:
  - `added_block`
  - `artistic_good`
  - `agriculture_product`, which has its own sub-types `crop` and `stored`
  - `building_component`
  - `building_functionality`
  - `business_operations`
  - `electrical_network_component`
  - `fixed_asset`, which has its own sub-types `infrastructure`, `land`, `machinery` and `material`
  - `livestock`
  - `network_service`
  - `product`
  - `provision`
  - `staff`
  - `tlc_network_component`
  - `transport_network_component`
- **Belongs to:** `enterprise` (A component is linked to one enterprise)
- **Has many:** `damage` (A component can be affected by multiple damage over time)

Table: `added_block`

Description (EN): This table represents added blocks of buildings. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the added block	`42`	PK, FK	No	Prima also a Foreig refere `compo`
`position`	VARCHAR	[...]	`EXTERNAL`	-	No	[...]
`vertical_mat`	VARCHAR	[...]	[...]	-	No	[...]
`horizontal_mat`	VARCHAR	[...]	[...]	-	No	[...]
`n_floor`	BIGINT	Number of floors of the added block	`2`	-	No	[...]
`total_height`	BIGINT	Total height of the added block in [measurement unit]	`200`	-	No	[...]
`floor_surface`	BIGINT	Total floor surface of the added block in [measurement unit]	`200`	-	No	[...]
`floor_height`	BIGINT	Total floor height of the added block in [measurement unit]	`200`	-	No	[...]
`function`	VARCHAR	[...]	[...]	-	No	[...]
`previous_survey`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `added_block` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry` , `town` , `country`), or ownership details of a added_block, you must join this table with the `component` record that shares the exact same `id` .

EXAMPLE QUERY

Retrieve a list of all added blocks, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    ab.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
```

```
FROM
    added_block ab
JOIN
    component c ON ab.id = c.id
WHERE
    c.component_kind  = 'ADDED_BLOCK';
```

Table: `artistic_good`

Description (EN): This table represents artistic goods. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the artistic good	`42`	PK, FK	No	Primar also ac Foreign referen `compon`
`kind`	VARCHAR	[...]]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `artistic_good` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of an artistic good, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all artistic good, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    ag.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    added_block ag
JOIN
    component c ON ag.id = c.id
WHERE
    c.component_kind  = 'ARTISTIC_GOOD';
```

Agriculture subclasses

TABLE: `agriculture_product`

Description (EN): This table represents agriculture products. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Com
`id`	BIGINT	Unique identifier for the agriculture product	`42`	PK, FK	No	Prim also Forei refer `comp`
`average_invested`	BIGINT	[...]	[...]	-	No	[...]
`expected_yield`	BIGINT	[...]	[...]	-	No	[...]
`gate_price`	BIGINT	[...]	[...]	-	No	[...]
`year_production`	BIGINT	[...]	[...]	-	No	[...]
`subtype`	VARCHAR	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `component`. The `agriculture_product` table is a specialized extension of the `component` table.
- **Has Sub-types (Inheritance):** The following tables inherit from `agriculture_product` and share its `id`:
  - `crop`
  - `stored`
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of an agriculture product, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all agriculture product, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    ap.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    agriculture_product ap
JOIN
    component c ON ap.id = c.id
WHERE
    c.component_kind  = 'AGRICULTURE_PRODUCT';
```

TABLE: `crop`

Description (EN): This table represents crops. In the database's inheritance architecture, it acts as a direct sub-type of the `agriculture_product` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for crop record	`42`	PK, FK	No	Primar as a Fo referen `agricu` and on `compor`
`area_planted`	BIGINT	[...]	[...]	-	No	[...]
`category`	VARCHAR	[...]	[...]	-	No	[...]
`growth`	VARCHAR	[...]	[...]	-	No	[...]
`duration`	VARCHAR	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `agriculture_product`. The `crop` table is a specialized extension of the `agriculture_product` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a crop, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all crop, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    c.id,
    c.area_planted,
    c.category,
    ap.year_production
FROM
    crop c join agriculture_product ap on c.id = ap.id
where ap.subtype = 'CROP'
```

TABLE: `stored`

Description (EN): This table represents stored. In the database's inheritance architecture, it acts as a direct sub-type of the `agriculture_product` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for stored agriculture products	`42`	PK, FK	No	Primar as a Fo referen `agricu` and on `compor`
`total_stored`	BIGINT	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `agriculture_product`. The `crop` table is a specialized extension of the `agriculture_product` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a stored product, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all stored records, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT s.* FROM public."stored" AS s
```

Table: `building_functionality`

Description (EN): This table represents building functionality of components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the building component functionality	`42`	PK, FK	No	Primar also ac Foreign referen `compon`
`floor_number`	BIGINT	Number of floors	`2`	-	No	-

[...]

Table: `building_component`

Description (EN): This table stores the structural, economic and physical characteristics of buildings, including construction details, material types and dimensions. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the building component	`42`	PK, FK	No
`principal`	BOOLEAN	Whether this is the principal building component	`true`	-	No
`subtype`	VARCHAR(255)	[...]	[...]	-	No
`public_service_code`	VARCHAR(255)	[...]	[...]	-	No
`heritage_tier`	VARCHAR(255)	[...]	[...]	-	No
`year`	INTEGER	Construction year	`1999`	-	No
`quality`	VARCHAR(255)	Overall quality	`GOOD`	-	No
`conservation`	VARCHAR(255)	Conservation status of the building	`POORLY_MANTAINED`	CHECK	No
`retrofitting`	VARCHAR(255)	Status of retrofitting	`NOT_IDENTIFIABLE`	CHECK	No
`regularity`	VARCHAR(255)	Regularity of structure	`NOT_IDENTIFIABLE`	CHECK	No
`inhabitants`	INTEGER	Number of inhabitants of the given building	`6`	-	No
`construction_cost`	INTEGER	Construction cost	`500000`	-	No
`economic_value`	INTEGER	Economic value	`1000000`	-	No
`vertical_mat`	VARCHAR(255)	Construction material of the vertical structures	`NOT_IDENTIFIABLE`	CHECK	No
`horizontal_mat`	VARCHAR(255)	Construction material of the horizontal structures	`MASONRY_ADOBE`	CHECK	No
`roof_mat`	VARCHAR(255)	Material of the roof element	`NOT_IDENTIFIABLE`	CHECK	No
`build_position`	VARCHAR(255)	Building position	`SINGLE_DETACHED`	CHECK	No

Column	Data type	Definition	Example value	Constraint?	Geometry?
`ground_floor`	VARCHAR(255)	Typology of ground floor	`OPEN`	CHECK	No
`n_floors`	INTEGER	Number of floors above the ground	`3`	-	No
`floor_height`	INTEGER	Average height of floors	`4`	-	No
`n_under_floors`	INTEGER	Number of underground floors	`1`	-	No
`floor_area`	INTEGER	Metric surface of the floor area	`1000`	-	No
`usage`	VARCHAR(255)	[...]	[...]	-	No
`foundation_typology`	VARCHAR(255)	Typology of the foundations	[...]	-	No
`main_use`	VARCHAR(255)	Main use of the building	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `building_component` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a building component, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all building components, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    bc.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    building_component bc
JOIN
    component c ON bc.id = c.id
WHERE
    c.component_kind = 'BUILDING_COMPONENT';
```

Table: `business_operations`

Description (EN): This table represents business operation records. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the business operation	`42`	PK, FK	No	Primar also ac Foreign referen `compon`

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `business_operations` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a business operation, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all business operations, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    bo.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    business_operations bo
JOIN
    component c ON bo.id = c.id
WHERE
    c.component_kind  = 'BUSINESS_OPERATIONS';
```

Table: `electrical_network_component`

Description (EN): This table represents electrical network components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the electrical network component	`42`	PK, FK	No	Primar also ac Foreign referen `compor`
`is_private`	BOOLEAN	[...]	[...]	-	No	[...]
`subtype`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `electrical_network_component` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of an electrical network component, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all electrical network component, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    en.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    electrical_network_component en
JOIN
    component c ON en.id = c.id
WHERE
    c.component_kind  = 'ELECTRICAL_COMPONENT';
```

Fixed assets subclasses

TABLE: `fixed_asset`

Description (EN): This table represents fixed asset components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the fixed asset component	`42`	PK, FK	No	Primar also ac Foreign referen `compon`

Relationships

- **Inherits from (Sub-type of):** `component`. The `fixed_asset` table is a specialized extension of the `component` table.
- **Has Sub-types (Inheritance):** The following tables inherit from `fixed_asset` and share its `id`:
  - `infrastructure`
  - `land`
  - `machinery`
  - `material`
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a fixed_asset, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all fixed asset, including their names, towns, and spatial coordinates, by joining the table with its parent `enterprise` entity:

```
SELECT
    fa.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    fixed_asset
JOIN
    component c ON fa.id = c.id
WHERE
    c.component_kind  = 'FIXED_ASSET';
```

TABLE: `infrastructure`

Description (EN): This table represents infrastructure. In the database's inheritance architecture, it acts as a direct sub-type of the `fixed_asset` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for infrastructure record	`42`	PK, FK	No	Primar also ac Foreign referen `fixed_` and on `compon`
`age`	BIGINT	[...]	[...]	-	No	[...]
`capacity`	BIGINT	[...]	[...]	-	No	[...]
`dimension`	BIGINT	[...]	[...]	-	No	[...]
`value`	BIGINT	[...]	[...]	-	No	[...]
`material`	VARCHAR	[...]	[...]	-	No	[...]
`function`	VARCHAR	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `fixed_asset`. The `infrastructure` table is a specialized extension of the `fixed_asset` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of an infrastructure record, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all infrastructure, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT i.* FROM public.infrastructure AS i
```

TABLE: `land`

Description (EN): This table represents land. In the database's inheritance architecture, it acts as a direct sub-type of the `fixed_asset` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for land records	`42`	PK, FK	No	Primar also ac Foreign referen `fixed_` and on `compon`
`dimension`	BIGINT	[...]	[...]	-	No	[...]
`value`	BIGINT	[...]	[...]	-	No	[...]
`soil`	VARCHAR	[...]	[...]	-	No	[...]
`configuration`	VARCHAR	[...]	[...]	-	No	[...]
`function`	VARCHAR	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `fixed_asset` . The `land` table is a specialized extension of the `fixed_asset` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry` , `town` , `country`), or ownership details of a land record, you must join this table with the `component` record that shares the exact same `id` .

Example Query

Retrieve a list of all lands, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT l.* FROM public.land AS l
```

TABLE: `machinery`

Description (EN): This table represents machinery. In the database's inheritance architecture, it acts as a direct sub-type of the `fixed_asset` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for machinery record	`42`	PK, FK	No	Primar also ac Foreign referen `fixed_` and on `compon`
`age`	BIGINT	[...]	[...]	-	No	[...]
`value`	BIGINT	[...]	[...]	-	No	[...]
`capacity`	BIGINT	[...]	[...]	-	No	[...]
`dimension`	VARCHAR	[...]	[...]	-	No	[...]
`material`	VARCHAR	[...]	[...]	-	No	[...]
`model`	VARCHAR	[...]	[...]	-	No	[...]
`function`	VARCHAR	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `fixed_asset` . The `machinery` table is a specialized extension of the `fixed_asset` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry` , `town` , `country`), or ownership details of a machinery record, you must join this table with the `component` record that shares the exact same `id` .

Example Query

Retrieve a list of all machinery, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT m.* FROM public.machinery AS m
```

TABLE: `material`

Description (EN): This table represents material. In the database's inheritance architecture, it acts as a direct sub-type of the `fixed_asset` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for material record	`42`	PK, FK	No	Primar also ac Foreign referen `fixed_` and on `compon`
`value`	BIGINT	[...]	[...]	-	No	[...]
`quantity`	VARCHAR	[...]	[...]	-	No	[...]
`material_type`	VARCHAR	[...]	[...]	-	No	[...]
`hazardous`	BOOLEAN	[...]	[...]	-	No	[...]

Relationships

- **Inherits from (Sub-type of):** `fixed_asset`. The `material` table is a specialized extension of the `fixed_asset` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a material record, you must join this table with the `component` record that shares the exact same `id`.

Example Query

Retrieve a list of all materials, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT m.* FROM public.material AS m
```

Table: `livestock`

Description (EN): This table represents livestock components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the livestock	`42`	PK, FK	No	Prima also a Foreig refere `compo`
`age`	BIGINT	[...]	`2`	-	No	-
`breeding_cost`	BIGINT	[...]	`2`	-	No	-
`gate_price`	BIGINT	[...]	`2`	-	No	-
`weight`	BIGINT	[...]	`2`	-	No	-
`year_production`	BIGINT	[...]	`2`	-	No	-
`subtype`	VARCHAR	[...]	-	-	No	-
`purpose`	VARCHAR	[...]	-	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `livestock` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a livestock, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all livestocks, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    l.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    livestock l
JOIN
    component c ON l.id = c.id
WHERE
    c.component_kind  = 'LIVESTOCK';
```

Table: `network_service`

Description (EN): This table represents network service components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the network service component	`42`	PK, FK	No	Primar also ac Foreign referen `compo`
`value`	BIGINT	[...]	[...]	-	No	-
`service`	VARCHAR	[...]	[...]	-	No	-
`service_user`	VARCHAR	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `network_service` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a network_service, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all network services, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    ns.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    network_service ns
JOIN
    component c ON ns.id = c.id
```

Table: `product`

Description (EN): This table represents business products. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for business product	`42`	PK, FK	No	Primar also ac Foreign referen `compon`
`product_kind`	VARCHAR	[...]	[...]	-	No	-
`category`	VARCHAR	[...]	[...]	-	No	-
`quantity`	BIGINT	[...]	[...]	-	No	-
`unit`	VARCHAR	[...]	[...]	-	No	-
`value`	BIGINT	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `product` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry` , `town` , `country`), or ownership details of a product, you must join this table with the `component` record that shares the exact same `id` .

EXAMPLE QUERY

Retrieve a list of all products, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    p.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    product p
JOIN
    component c ON p.id = c.id
```

Table: `provision`

Description (EN): This table represents provision components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the provision component	`42`	PK, FK	No	Primar also ac Foreign referen `compon`
`value`	BIGINT	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `provision` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a provision, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all provisions, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    p.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    provision p
JOIN
    component c ON p.id = c.id
```

Table: `staff`

Description (EN): This table represents staff components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the staff component	`42`	PK, FK	No	Primar also ac Foreign referen `compor`
`number`	BIGINT	[...]	[...]	-	No	-
`activities`	VARCHAR	[...]	[...]	-	No	-
`staff_category`	VARCHAR	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `staff` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a staff, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all staff records, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    s.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    staff s
JOIN
    component c ON s.id = c.id
```

Table: `transport_network_component`

Description (EN): This table represents transport network components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the transport network component	`42`	PK, FK	No	Primar also ac Foreign referen `compor`
`is_private`	BOOLEAN	[...]	[...]	-	No	-
`lanes_number`	BIGINT	[...]	[...]	-	No	-
`subtype`	VARCHAR	[...]	[...]	-	No	Possib PRIMA PROVI COUNT

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `transport_network_component` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or ownership details of a transport network component, you must join this table with the `component` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all transport network components, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
 tnc2.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    transport_network_component tnc2
JOIN
    component c ON tnc2.id = c.id
```

Table: `tlc_network_component`

Description (EN): This table represents telecommunication network components. In the database's inheritance architecture, it acts as a direct sub-type of the `component` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `component` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the telecommunication network component	`42`	PK, FK	No	Prima also Forei refer `comp`
`value`	BIGINT	[...]	[...]	-	No	-
`component`	VARCHAR	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `component`. The `tlc_network_component` table is a specialized extension of the `component` table.
- **Belongs to:** `enterprise` (A component sub-type is linked to one enterprise)
- **Has many:** `damage` (A component sub-type can be affected by multiple damage over time)

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry` , `town` , `country`), or ownership details of a telecommunication network component, you must join this table with the `component` record that shares the exact same `id` .

EXAMPLE QUERY

Retrieve a list of all telecommunication network components, including their names, towns, and spatial coordinates, by joining the table with its parent `component` entity:

```
SELECT
    tnc.id,
    c.name,
    c.town,
    ST_AsText(c.geometry) as coordinates
FROM
    tlc_network_component tnc
JOIN
    component c ON tnc.id = c.id
```

3.1.4 Table: `attachment`

Description (EN): This table stores the information and details related to an attachment.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the attachment	`3`	PK	No	[...]
`damage_id`	BIGINT	Foreign key referring to a reported damage	[...]	FK	No	Pointin entity c
`date`	TIMESTAMP	[...]	-	-	No	-
`size`	BIGINT	[...]	-	-	No	-
`description`	VARCHAR	[...]	-	-	No	-
`name`	VARCHAR	[...]	-	-	No	-
`type`	VARCHAR	[...]	-	-	No	-
`data`	BIGINT	[...]	-	-	No	-

Relationships

- **Documents** `damage`

3.1.5 Table: `survey`

Description (EN): This table stores the information and details related to a specific damage assessement survey conducted following a major event.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the survey	`3`	PK	No	[...]
`created_on`	TIMESTAMP	Time of execution of the assessment survey	[...]	NOT NULL	No	[...]
`enterprise_id`	BIGINT	Foreign key referring to an enterprise	[...]	FK	No	Pointin entity e
`event_id`	BIGINT	Time of execution of the assessment survey	[...]	FK	No	Pointin entity e

Relationships

Because `survey` is one of the core entities of the schema, it has extensive relationships:

- **Reports** `damage`
- **Is linked to** `enterprise`
- **Follows an** `event`

3.1.6 Table: `event`

Description (EN): This table stores the information and details related to a specific major event.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the event	`3`	PK	No
`date`	TIMESTAMP	Time of occurrence of the event	[...]	-	No
`date_end`	TIMESTAMP	Ending time of occurrence of the event	[...]	-	No
`parent_event_id`	BIGINT	Reference in case of multi-event	[...]	FK	No
`name`	VARCHAR	Name of the event	`Storm Example`	-	No
`severity`	VARCHAR	Severity level of the event	[...]	-	No
`type`	VARCHAR	Type of the event	[...]	-	No
`country`	VARCHAR	Country where the event is located	`Italy`	-	No
`state`	VARCHAR	State or region where the event is located	`Lazio`	-	No
`county`	VARCHAR	County or province where the event is located	`Rome`	-	No
`town`	VARCHAR	Municipality where the event is located	`Rome`	-	No
`hamlet`	VARCHAR	Hamlet or locality where the event is located	`Trastevere`	-	No
`road`	VARCHAR	Street where the event is located	`Via Appia`	-	No
`house_number`	VARCHAR	House number of the event	`10`	-	No
`postcode`	VARCHAR	Postal code of the event	`00100`	-	No
`geometry`	GEOMETRY	Spatial representation of the event	[...]	-	Yes
`coordinates_inferred`	BOOLEAN	Whether the coordinates of the event were inferred rather	[...]	-	No

Column	Data type	Definition	Example value	Constraint?	Geometry?
		than directly measured			

Relationships

- **Has many:** `survey` (An event can be followed by multiple damage assessment surveys)
- **Is part of / has parts:** `event` through `parent_event_id`. A multi-event can be split into child events that point to the parent event.

3.1.7 Damage and subclasses

Table: `damage`

Description (EN): This table stores the universal, high-level attributes—such as identification and geographical location for all damages following the occurrance of an event. Specific types of damages (like damages to artistic goods, damages on building functionality, or other) store their specialized data in separate sub-tables that share the same ID.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage	`3`	PK	No
`estimated_value`	BIGINT	[...]	-	-	No
`component_id`	BIGINT	[...]	-	FK	No
`date`	TIMESTAMP	[...]	[...]	-	No
`end_date`	TIMESTAMP	[...]	[...]	-	No
`start_end`	TIMESTAMP	[...]	[...]	-	No
`survey_id`	BIGINT	[...]	[...]	FK	No
`country`	VARCHAR	[...]	[...]	-	No
`county`	VARCHAR	[...]	[...]	-	No
`description`	VARCHAR	[...]	[...]	-	No
`hamlet`	VARCHAR	[...]	[...]	-	No
`house_number`	VARCHAR	[...]	[...]	-	No
`postcode`	VARCHAR	[...]	[...]	-	No
`road`	VARCHAR	[...]	[...]	-	No
`state`	VARCHAR	[...]	[...]	-	No
`town`	VARCHAR	[...]	[...]	-	No
`geometry`	GEOMETRY	Spatial representation of the damage entity	[...]	-	No
`coordinates_inferred`	BOOLEAN	[...]	[...]	-	No
`sendai_indicator`	VARCHAR	[...]	[...]	-	No

RELATIONSHIPS

Because `damage` is one of the core entities of the schema, it has extensive relationships:

- **Has Sub-types (Inheritance):** The following tables inherit from `damage` and share its `id`:
  - `damage_artistic_good`

    - `damage_building_functionality`
    - `damage_building_not_structural`
    - `damage_building_structural`
    - `damage_business_operations`
    - `damage_church_structure`
    - `damage_goods`
    - `damage_network_component`
    - `damage_network_service`
    - `damage_people`
    - `damage_product`
    - `damage_vehicles`
    - `damage_whole_structure`
    - `farm_damage`
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is recorded by:** `survey` (One or more damages is reported in a survey)
- **Documented in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

Table: `damage_artistic_good`

Description (EN): This table represents damages to artistic goods. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Con
`id`	BIGINT	Unique identifier for the damage to artistic goods	`42`	PK, FK	No	Prin also Fore refe `dam`
`kind`	VARCHAR	[...]	[...]	-	No	[...]
`level`	VARCHAR	[...]	[...]	-	No	[...]
`extension_percent`	BIGINT	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_artistic_good` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to artistic good, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to artistic goods, including their names, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dag.*,
    d.*
FROM
    damage_artistic_good dag join damage d on dag.id = d.id
```

Table: `damage_building_functionality`

Description (EN): This table represents damages to building functionalities. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage to a building functionality	`42`	PK, FK	No
`floor_number`	BIGINT	[...]	[...]	-	No
`loss_of_service`	BIGINT	[...]	[...]	-	No
`temporary_description`	VARCHAR	[...]	[...]	-	No
`temporary`	VARCHAR	[...]	[...]	-	No
`outage`	VARCHAR	[...]	[...]	-	No
`subtype`	VARCHAR	[...]	[...]	-	No
`apartment_code`	VARCHAR	[...]	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_building_functionality` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to building functionalities, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to building functionalities, including their names, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dbf.*,
    d.*
FROM
    damage_building_functionality dbf join damage d on dbf.id = d.id
```

Table: `damage_building_not_structural`

Description (EN): This table represents non-structural damages to building functionalities. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the non-structural damage to a building	`42`	PK, FK	No
`non_structural_element`	VARCHAR	[...]	[...]	-	No
`non_structural_damage_type`	VARCHAR	[...]	[...]	-	No
`damage_quantification`	VARCHAR	[...]	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_building_not_structural` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a non-structural damage to buildings, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all non-structural damages to buildings, including their names, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dbns.*,
    d.*
FROM
    damage_building_not_structural dbns join damage d on dbns.id = d.id
```

Table: `damage_building_structural`

Description (EN): This table represents structural damages to building functionalities. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the structural damage to a building	`42`	PK, FK	No	Primar also ac Foreign referen `damage`
`extension`	BIGINT	[...]	[...]	-	No	[...]
`level`	VARCHAR	[...]	[...]	-	No	[...]
`classification`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_building_structural` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a structural damage to buildings, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all structural damages to buildings, including their names, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dbs.*,
    d.*
FROM
    damage_building_structural dbs  join damage d on dbs.id = d.id
```

Table: `damage_business_operations`

Description (EN): This table represents structural damages to business operations. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Com
`id`	BIGINT	Unique identifier for the damage to business operations	`42`	PK, FK	No	Prim also Forei refer `dama`
`disruption_type`	VARCHAR	[...]	[...]	-	No	[...]
`disruption_level`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_business_operations` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to business operations, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to business operations, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dbo.*,
    d.*
FROM
    damage_business_operations dbo    join damage d on dbo.id = d.id
```

Table: `damage_church_structure`

Description (EN): This table represents structural damages to church structures. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage to church structures	`42`	PK, FK	No
`variant`	VARCHAR	[...]	[...]	-	No
`scale`	VARCHAR	[...]	[...]	-	No
`extension_percent`	BIGINT	[...]	[...]	-	No
`damage_to_foundations`	BOOLEAN	[...]	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_church_structure` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to church structures, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to goods, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dcs.*,
    d.*
FROM
    damage_church_structure dcs   join damage d on dcs.id = d.id
```

Table: `damage_goods`

Description (EN): This table represents damages to goods. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the damage to goods	`42`	PK, FK	No	Primar also ac Foreign referen `damage`
`quantity`	BIGINT	[...]	[...]	-	No	[...]
`good`	VARCHAR	[...]	[...]	-	No	[...]
`unit`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_goods` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to goods, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to goods, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dg.*,
    d.*
FROM
    damage_goods dg  join damage d on dg.id = d.id
```

Table: `damage_network_component`

Description (EN): This table represents structural damages to network component. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage to network component	`42`	PK, FK	No
`reachable`	BOOLEAN	[...]	[...]	-	No
`network_damage_level`	VARCHAR	[...]	[...]	-	No
`electrical_damage_category`	VARCHAR	[...]	[...]	-	No
`transport_damage_category`	VARCHAR	[...]	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_network_component` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to network components, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to network components, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dnc.*,
    d.*
FROM
    damage_network_component dnc   join damage d on dnc.id = d.id
```

Table: `damage_network_service`

Description (EN): This table represents structural damages to network service. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Con
`id`	BIGINT	Unique identifier for the damage to network service	`42`	PK, FK	No	Prin also Fore refe `dam`
`affected_users`	BIGINT	[...]	[...]	-	No	[...]
`loss_of_service`	BIGINT	[...]	[...]	-	No	[...]
`electrical_outage`	VARCHAR	[...]	[...]	-	No	[...]
`telecom_outage`	VARCHAR	[...]	[...]	-	No	[...]
`transport_outage`	VARCHAR	[...]	[...]	-	No	[...]

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_network_service` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to network service, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to network service, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dns.*,
    d.*
FROM
    damage_network_service dns join damage d on dns.id = d.id
```

Table: `damage_people`

Description (EN): This table represents damages reported to people. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Con
`id`	BIGINT	Unique identifier for damages to people	`42`	PK, FK	No	Prin also Fore refe `dam`
`number_of_dead`	INTEGER	[...]	[...]	-	No	-
`number_of_injured`	INTEGER	[...]	[...]	-	No	-
`number_of_missing`	INTEGER	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_people` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to people, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to people, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dp.*,
    d.*
FROM
    damage_people dp  join damage d on dp.id = d.id
```

Table: `damage_product`

Description (EN): This table represents damages to products. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for damages to products	`42`	PK, FK	No
`quality_loss_percentage`	INTEGER	[...]	[...]	-	No
`manufacturing_loss_percentage`	INTEGER	[...]	[...]	-	No
`total_damage_percentage`	INTEGER	[...]	[...]	-	No
`non_produced_good_quantity`	INTEGER	[...]	[...]	-	No
`non_transformed_good_quantity`	INTEGER	[...]	[...]	-	No

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_product` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to products, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to products, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dp.*,
    d.*
FROM
    damage_product dp  join damage d on dp.id = d.id
```

Table: `damage_vehicles`

Description (EN): This table represents damages reported to vehicles. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the whole structure damage	`42`	PK, FK	No	Primar also ac Foreign referen `damage`
`quantity`	INTEGER	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_vehicles` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to vehicles, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to vehicles, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dv.*,
    d.*
FROM
    damage_vehicles dv join damage d on dv.id = d.id
```

Table: `damage_whole_structure`

Description (EN): This table represents damages reported to whole structures. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as name, location, management type, and geometry) from its parent `damage` record.

COLUMN STRUCTURE

Column	Data type	Definition	Example value	Constraint?	Geometry?	Con
`id`	BIGINT	Unique identifier for the whole structure damage	`42`	PK, FK	No	Prin also Fore refe `dam`
`scale`	VARCHAR	[...]	[...]	-	No	-
`extension_percent`	INTEGER	[...]	[...]	-	No	-

RELATIONSHIPS

- **Inherits from (Sub-type of):** `damage`. The `damage_whole_structure` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the name, geographical location (`geometry`, `town`, `country`), or other details of a damage to the whole structure, you must join this table with the `damage` record that shares the exact same `id`.

EXAMPLE QUERY

Retrieve a list of all damages to whole structure, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dws.*,
    d.*
FROM
    damage_whole_structure dws join damage d on dws.id = d.id
```

Damage to farm and subclasses

TABLE: `farm_damage`

Description (EN): This table represents damages reported to farms. In the database's inheritance architecture, it acts as a direct sub-type of the `damage` table. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the damage to a farm	`42`	PK, FK	No	Primar also ac Foreign referen `damage`

Relationships

- **Inherits from (Sub-type of):** `damage`. The `farm_damage` table is a specialized extension of the `damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`
- **Has Sub-types (Inheritance):** The following tables inherit from `farm_damage` and share its `id`:
  - `damage_agriculture_product`
  - `damage_fixed_asset`
  - `damage_livestock`
  - `damage_provision`
  - `damage_staff`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to farms, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to farms, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    fd.*,
    d.*
FROM
    farm_damage fd JOIN damage d ON fd.id = d.id;
```

TABLE: `damage_agriculture_product`

Description (EN): This table represents damages reported to agriculture products. In the database's inheritance architecture, it acts as a direct sub-type of the `farm_damage` table, which is itself a sub-type of `damage`. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage to an agriculture product	`42`	PK, FK	No
`non_produced_good_quantity`	DOUBLE PRECISION	Quantity of goods not produced because of the damage	[...]	-	No
`quality_loss_percentage`	DOUBLE PRECISION	Percentage of quality loss of the product	[...]	-	No
`total_damage_percentage`	DOUBLE PRECISION	Total percentage of damage to the product	[...]	-	No
`total_losses_percentage`	DOUBLE PRECISION	Total percentage of losses	[...]	-	No
`yield_loss_percentage`	DOUBLE PRECISION	Percentage of yield loss	[...]	-	No

Relationships

- **Inherits from (Sub-type of):** `farm_damage`, which in turn inherits from `damage`. The `damage_agriculture_product` table is a specialized extension of the `farm_damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to agriculture products, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to agriculture products, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dap.*,
    d.*
FROM
    damage_agriculture_product dap JOIN damage d ON dap.id = d.id;
```

TABLE: `damage_fixed_asset`

Description (EN): This table represents damages reported to farm fixed assets. In the database's inheritance architecture, it acts as a direct sub-type of the `farm_damage` table, which is itself a sub-type of `damage`. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the damage to a farm fixed asset	`42`	PK, FK	No	Primar also ac Foreign referen `farm_` (and, in `damage`
`perc_area`	DOUBLE PRECISION	Percentage of the area affected	[...]	-	No	-
`size`	DOUBLE PRECISION	Size of the affected fixed asset	[...]	-	No	-

Relationships

- **Inherits from (Sub-type of):** `farm_damage`, which in turn inherits from `damage`. The `damage_fixed_asset` table is a specialized extension of the `farm_damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to farm fixed assets, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to farm fixed assets, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dfa.*,
    d.*
FROM
    damage_fixed_asset dfa JOIN damage d ON dfa.id = d.id;
```

TABLE: `damage_livestock`

Description (EN): This table represents damages reported to livestock. In the database's inheritance architecture, it acts as a direct sub-type of the `farm_damage` table, which is itself a sub-type of `damage`. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	C
`id`	BIGINT	Unique identifier for the damage to livestock	`42`	PK, FK	No	P al F re `f` (a `d`
`number_of_dead`	INTEGER	Number of animals dead	[...]	-	No	-
`number_of_diseased`	INTEGER	Number of animals diseased	[...]	-	No	-
`number_of_injured`	INTEGER	Number of animals injured	[...]	-	No	-
`number_of_missing`	INTEGER	Number of animals missing	[...]	-	No	-
`disease_description`	VARCHAR	Description of the disease	[...]	-	No	-

Relationships

- **Inherits from (Sub-type of):** `farm_damage`, which in turn inherits from `damage`. The `damage_livestock` table is a specialized extension of the `farm_damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to livestock, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to livestock, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dl.*,
    d.*
FROM
    damage_livestock dl JOIN damage d ON dl.id = d.id;
```

TABLE: `damage_provision`

Description (EN): This table represents damages reported to provisions. In the database's inheritance architecture, it acts as a direct sub-type of the `farm_damage` table, which is itself a sub-type of `damage`. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?
`id`	BIGINT	Unique identifier for the damage to provisions	`42`	PK, FK	No
`additional_food`	INTEGER	Additional cost of food	[...]	-	No
`additional_veterinary`	INTEGER	Additional veterinary cost	[...]	-	No
`losses_fees`	INTEGER	Losses in fees	[...]	-	No
`losses_premature_sale`	INTEGER	Losses from premature sale	[...]	-	No
`losses_value`	INTEGER	Value of the losses	[...]	-	No
`relocation_cost`	INTEGER	Cost of relocation	[...]	-	No
`replanting_cost`	INTEGER	Cost of replanting	[...]	-	No
`additional_materials`	INTEGER	Cost of additional materials	[...]	-	No
`restoration_cost`	INTEGER	Cost of restoration	[...]	-	No

Relationships

- **Inherits from (Sub-type of):** `farm_damage`, which in turn inherits from `damage`. The `damage_provision` table is a specialized extension of the `farm_damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to provisions, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to provisions, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    dp.*,
    d.*
FROM
    damage_provision dp JOIN damage d ON dp.id = d.id;
```

TABLE: `damage_staff`

Description (EN): This table represents damages reported to farm staff. In the database's inheritance architecture, it acts as a direct sub-type of the `farm_damage` table, which is itself a sub-type of `damage`. It does not store redundant descriptive columns; instead, it uses a shared primary key to inherit all high-level attributes (such as description, dates, location and geometry) from its parent `damage` record.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	C
`id`	BIGINT	Unique identifier for the damage to farm staff	`42`	PK, FK	No	P a F re `f` (a `d`
`duration_of_absence`	INTEGER	Duration of the absence of the staff	[...]	-	No	-
`number_of_absent`	INTEGER	Number of staff members absent	[...]	-	No	-
`number_of_dead`	INTEGER	Number of staff members killed	[...]	-	No	-
`number_of_injured`	INTEGER	Number of staff members injured	[...]	-	No	-
`further_details`	VARCHAR	Further details on the damage to the staff	[...]	-	No	-
`reason_of_absence`	VARCHAR	Reason for the absence	[...]	-	No	-
`number_of_missing`	INTEGER	Number of staff members missing	[...]	-	No	-

Relationships

- **Inherits from (Sub-type of):** `farm_damage`, which in turn inherits from `damage`. The `damage_staff` table is a specialized extension of the `farm_damage` table.
- **Affects:** `component` (One or more damages affect a component or its subclasses)
- **Is reported by:** `survey` (One or more damages is reported in a survey)
- **Reported in:** `attachment` (A damage is reported in one or more attachments)
- **Results in:** `economic_loss`

!!! tip "Where is the rest of the data?" To find the description, dates, geographical location (`geometry`, `town`, `country`), or other details of a damage to staff, you must join this table with the `damage` record that shares the exact same `id`.

Example Query

Retrieve a list of all damages to staff, including their descriptions, towns, and spatial coordinates, by joining the table with its parent `damage` entity:

```
SELECT
    ds.*,
    d.*
FROM
    damage_staff ds JOIN damage d ON ds.id = d.id;
```

3.1.8 Table: `economic_loss`

Description (EN): This table stores the costs associated with a reported damage: the estimated, approved and final amounts.

Column Structure

Column	Data type	Definition	Example value	Constraint?	Geometry?	Comm
`id`	BIGINT	Unique identifier for the economic loss record	`3`	PK	No	[...]
`damage_id`	BIGINT	Foreign key referring to a reported damage	[...]	FK	No	Pointin entity c
`approved_cost`	DOUBLE PRECISION	Cost approved for the damage	-	-	No	-
`estimated_cost`	DOUBLE PRECISION	Estimated cost of the damage	-	-	No	-
`final_cost`	DOUBLE PRECISION	Final cost of the damage	-	-	No	-
`type_of_cost`	VARCHAR	Type of cost	-	-	No	-

Relationships

- **Refers to:** `damage` through `damage_id`. The economic loss is the monetary result of a reported damage.

Example Query

Retrieve the costs recorded for each damage:

```
SELECT
    d.id AS damage_id,
    el.type_of_cost,
    el.estimated_cost,
    el.approved_cost,
    el.final_cost
FROM
    economic_loss el
JOIN
    damage d ON el.damage_id = d.id;
```

4. Contribution Guidelines

Thank you for helping us keep the Data Dictionary accurate and up-to-date!

Good documentation is a team effort. You do not need to be a developer, know how to write code, or understand Git to contribute to this project. All content updates are managed through a simple spreadsheet.

4.1 The Workflow

Updating the dictionary takes just three simple steps:

1. **Download the template:** Grab the latest version of the Data Dictionary Excel template from our shared team folder.
2. **Fill in the data:** Add or update the rows corresponding to the database tables and columns you are documenting.
3. **Submit for publishing:** Send the updated Excel file to the technical team. We will run an automated process to convert your spreadsheet into the web pages you see here.

4.2 How to fill out the Excel template

Each sheet in the Excel file represents a single database table. For every column in that table, please provide the following details:

- **Column:** The exact technical name of the field in the database (e.g., `user_id`, `geom`, `created_at`).
- **Data type:** The format of the data (e.g., `VARCHAR(255)`, `UUID`, `POINT`, `INTEGER`).
- **Definition (IT) & Definition (EN):** A clear, business-friendly description of what the column represents. Please fill out both the Italian and English columns to support our bilingual platform.
- **Example value:** A realistic example of the data to help others understand the format (e.g., `john.doe@email.com` or `POINT(12.49 41.89)`).
- **Constraint?:** Note any database rules, such as `PK` (Primary Key), `FK` (Foreign Key), or `NOT NULL`.
- **Geometry?:** Specify if the field contains spatial data (e.g., `Yes - EPSG:4326` or `No`).
- **Comments (IT) & Comments (EN):** Any extra context, business logic, or warnings regarding how this data should be used or interpreted.

!!! warning "Keep Formatting Simple" Please avoid using complex Excel formatting like merged cells, custom cell colors, or multiple lines inside a single cell. Plain text ensures the smoothest conversion to the web platform!

4.3 Submitting Your Changes

Once your Excel file is ready and reviewed:

- Upload the new version to the designated shared drive.
- Notify the database administration team via email or our internal chat.
- The team will process the file, and your changes will be live on the Data Dictionary within a few minutes!

5. Graphic User Interface terminology

The web-app application of AHEAD adopts aliases for the entities involved in the user form compilations. To improve understanding of the underlying connections to the database, this section includes a table mapping GUI interface terms to entity names in the PostgreSQL database, to support replicability and transparency of the system.

5.1 Backend to web-app correspondence

Entity name in the database	Website alias
`building`	Built environment
`business`	Economic Activities - Business
`farm`	Economic Activities - Farm
`electrical_system`	Lifelines - Electrical
`transport_system`	Lifelines - Transport
`tlc_system`	Lifelines - Telecommunication
`owner`	Owner profiles
`component`	Elements & production (all subclasses)
`added_block`	Added block
`building_component`	Principal structure
`fixed_asset`	Elements
`material`	Stocks - Stored inputs
`network_service`	Telecom service/Electrical service/Transport service
`transport_network_component`	Transport element
`tlc_network_component`	Telecom element