<https://gemini.google.com/u/1/app/cc0bffb89378d0df?is_sa=1&android-min-version=301356232&ios-min-version=322.0&campaign_id=bkws&utm_source=google&utm_medium=cpc&utm_campaign=2024enUS_gemfeb&pt=9008&mt=8&ct=p-growth-sem-bkws>

<https://gemini.google.com/u/1/app/a3fdff8118a11194>

### 3\. Updated Production-Ready Specification: `public.town\_types\_master`

1\. Purpose & Primary Use-Cases

This table serves as a master lookup for categorizing towns and settlements. It provides unique codes (e.g., 'city\_large', 'village\_paese') and associated non-translatable metadata like icons or display order. All display names and descriptions for these town types are managed in the `public.translations` table, ensuring consistent and translatable classification. It is referenced by `public.towns.town\_type\_code`.

Key User-Story Touchpoints:

- Pilgrim: Sees the translated town type on a town's page.

- Admin/Manager: Uses codes to classify towns; manages primary language names/descriptions in `translations`.

- Platform: Uses codes for consistent categorization.

2\. Schema

| Column Name | Data Type | Constraints | Description |

| `code` | `TEXT` | `PRIMARY KEY` | Unique, stable code for the town type (e.g., 'city\_large'). Used as `row\_foreign\_key` in `translations`. |

| `icon\_identifier` | `TEXT` | | Optional identifier for a UI icon (e.g., 'icon-city'). |

| `display\_order` | `INTEGER` | `NOT NULL DEFAULT 0`, `UNIQUE` | For ordering types in UI. Ensures deterministic sorting. |

| `is\_active` | `BOOLEAN` | `NOT NULL DEFAULT true` | Whether the town type is active and usable. |

| `created\_at` | `TIMESTAMPTZ` | `NOT NULL DEFAULT now()` | Timestamp of creation. |

| `updated\_at` | `TIMESTAMPTZ` | `NOT NULL DEFAULT now()` | Timestamp of last update. Auto-updated by trigger. |

| `created\_by\_profile\_id` | `UUID` | `REFERENCES public.profiles(id) ON DELETE SET NULL` | Profile ID of the user who created the record. |

| `updated\_by\_profile\_id` | `UUID` | `REFERENCES public.profiles(id) ON DELETE SET NULL` | Profile ID of the user who last updated the record. |

Translatable Fields (Managed in public.translations table):

The following conceptual fields for a town type will have their content stored in public.translations with table\_identifier = 'town\_types\_master' and row\_foreign\_key = town\_types\_master.code:

- `name` (e.g., `column\_identifier = 'name'`)

- `description` (e.g., `column\_identifier = 'description'`)

3\. PostgreSQL DDL

SQL

```

CREATE TABLE IF NOT EXISTS public.town\_types\_master (

code TEXT PRIMARY KEY,

icon\_identifier TEXT,

display\_order INTEGER NOT NULL DEFAULT 0 UNIQUE,

is\_active BOOLEAN NOT NULL DEFAULT true,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

created\_by\_profile\_id UUID REFERENCES public.profiles(id) ON DELETE SET NULL,

updated\_by\_profile\_id UUID REFERENCES public.profiles(id) ON DELETE SET NULL

);

COMMENT ON TABLE public.town\_types\_master IS 'Master lookup for categorizing towns and settlements. Names/descriptions are in public.translations. Referenced by towns.town\_type\_code. Version: 2.0';

COMMENT ON COLUMN public.town\_types\_master.code IS 'Primary Key. Unique code for the town type (e.g., ''city\_large'', ''village\_paese''). Short, stable, non-changing. Used as row\_foreign\_key in translations.';

COMMENT ON COLUMN public.town\_types\_master.icon\_identifier IS 'Optional identifier for a UI icon associated with this town type (e.g., ''icon-city'', ''icon-village'').';

COMMENT ON COLUMN public.town\_types\_master.display\_order IS 'Optional: for ordering types in dropdowns or lists. Unique if used for deterministic sorting.';

COMMENT ON COLUMN public.town\_types\_master.is\_active IS 'Indicates if the town type is active and available for use.';

COMMENT ON COLUMN public.town\_types\_master.created\_at IS 'Timestamp of when the town type record was created.';

COMMENT ON COLUMN public.town\_types\_master.updated\_at IS 'Timestamp of when the town type record was last updated. Auto-updated by a trigger.';

COMMENT ON COLUMN public.town\_types\_master.created\_by\_profile\_id IS 'Profile ID of the user who created the town type record.';

COMMENT ON COLUMN public.town\_types\_master.updated\_by\_profile\_id IS 'Profile ID of the user who last updated the town type record.';

-- Indexing:

-- PRIMARY KEY on 'code' already creates an index. [cite: 429]

-- UNIQUE constraint on 'display\_order' already creates an index. [cite: 429]

-- Trigger for updated\_at (ensure public.handle\_updated\_at() or equivalent function exists)

CREATE TRIGGER trigger\_town\_types\_master\_set\_updated\_at

BEFORE UPDATE ON public.town\_types\_master

FOR EACH ROW

EXECUTE FUNCTION public.handle\_updated\_at(); -- Or extensions.moddatetime, or public.set\_current\_timestamp\_updated\_at [cite: 430]

-- Trigger for deleting related translations

-- (ensure public.cleanup\_related\_translations() function is adaptable for TEXT PKs

-- or a specific version is made that correctly uses OLD.code as the row\_foreign\_key

-- for table\_identifier = 'town\_types\_master')

CREATE TRIGGER trigger\_delete\_town\_type\_translations

AFTER DELETE ON public.town\_types\_master

FOR EACH ROW EXECUTE FUNCTION public.cleanup\_related\_translations(); -- [cite: 431] Function needs to handle OLD.code

-- RLS Policies

ALTER TABLE public.town\_types\_master ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Allow admin full access on town\_types\_master"

ON public.town\_types\_master

FOR ALL

USING (public.is\_platform\_admin()) -- Assumes helper function

WITH CHECK (public.is\_platform\_admin());

CREATE POLICY "Allow authenticated users read access on town\_types\_master"

ON public.town\_types\_master

FOR SELECT

TO authenticated

USING (is\_active = true);

CREATE POLICY "Allow anonymous users read access on town\_types\_master"

ON public.town\_types\_master

FOR SELECT

TO anon

USING (is\_active = true);

```

4\. JSON Schema Mirror

JSON

```

{

"title": "town\_type\_master",

"description": "Master lookup for categorizing towns and settlements. Names/descriptions are in 'translations' table. Version: 2.0",

"type": "object",

"properties": {

"code": {

"type": "string",

"description": "Primary Key. Unique code for the town type (e.g., 'city\_large'). Used as 'row\_foreign\_key' in 'translations'.",

"pattern": "^[a-z0-9\_]+$"

},

"icon\_identifier": {

"type": ["string", "null"],

"description": "Optional identifier for a UI icon associated with this town type.",

"maxLength": 50

},

"display\_order": {

"type": "integer",

"default": 0,

"description": "Order for displaying types in UI. Must be unique."

},

"is\_active": {

"type": "boolean",

"default": true,

"description": "Indicates if the town type is active and available for use."

},

"created\_at": {

"type": "string",

"format": "date-time",

"readOnly": true

},

"updated\_at": {

"type": "string",

"format": "date-time",

"readOnly": true

},

"created\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID of the user who created the record."

},

"updated\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID of the user who last updated the record."

}

},

"required": [

"code",

"display\_order",

"is\_active"

]

}

```

5\. Relationships & Integrity

- Primary Key: `code` (TEXT).

- Referenced By:

- `public.towns.town\_type\_code` REFERENCES `public.town\_types\_master(code)` ON DELETE SET NULL (or RESTRICT).

- `public.translations.row\_foreign\_key` (conceptually, when `table\_identifier = 'town\_types\_master'`). Managed by trigger and application logic.

- Unique Constraints: `code` (PK), `display\_order`.

6\. Multilingual Strategy

All display names and descriptions for town types are stored in the public.translations table. The town\_types\_master table itself holds no direct user-facing textual name or description. To get the name of a town type:

SQL

```

SELECT translated\_text

FROM public.translations

WHERE table\_identifier = 'town\_types\_master'

AND column\_identifier = 'name' -- Or 'description'

AND row\_foreign\_key = 'type\_code\_value' -- e.g., 'village\_paese' [cite: 441]

AND language\_code = 'user\_language\_code'

AND translation\_status = 'published\_live'; -- Or relevant status [cite: 441]

```

7\. Role-Based Workflow & RLS Notes

- Content Management: Core records (`code`, `icon\_identifier`, `display\_order`, `is\_active`) managed by Administrators. Textual content via `public.translations` workflow.

- RLS Policies: Admins full access (via `public.is\_platform\_admin()`). Authenticated/Anonymous read access to `is\_active = true` types. `public.translations` RLS governs text access.

8\. ENUM vs Lookup Discussion

This table is the lookup table, replacing an ENUM for richer functionality and i18n.

9\. UI/UX Enablement

- UI uses `code` for logic/translation fetching. `icon\_identifier` and `display\_order` for presentation.

- Translated `name`/`description` from `public.translations` are displayed.

- Enables filtering towns by type, displaying categorized lists.

10\. Key Considerations & Definitions

- `code` Stability: `code` values must be stable as they are PKs and FKs.

- Data Entry Workflow: Create record in `town\_types\_master`, then add corresponding `name`/`description` in `public.translations` for primary and other languages.

11\. Scalability & Future-Proofing

Highly scalable for languages and flexible for adding non-translatable attributes. The number of town types is expected to be small.

12\. Next-Action Checklist

- 🔴 Define Helper Function: Ensure `public.is\_platform\_admin()` is defined and secure.

- 🔴 Define `updated\_at` Trigger Function: Ensure standard `public.handle\_updated\_at()` is defined.

- 🔴 Verify/Adapt `cleanup\_related\_translations`: Confirm function handles `TEXT` PKs (`OLD.code`) for `town\_types\_master`.

- 🟠 Deploy DDL: Apply `CREATE TABLE` and `COMMENT` DDL for `public.town\_types\_master`.

- 🟠 Populate Initial Data: Insert seed data (e.g., `{'city\_large', 'landmark-city', 10, true}`).

- 🟠 Populate Translations: For each seed `code`, create `name`/`description` entries in `public.translations`.

- 🟠 Implement Triggers: Create `trigger\_town\_types\_master\_set\_updated\_at` and `trigger\_delete\_town\_type\_translations`.

- 🟠 Implement RLS Policies: Apply defined RLS policies.

- 🟢 Update `towns` Table FK: Ensure `towns.town\_type\_code` correctly references this table.

- 🟢 Adapt Data Access Layer: Ensure application fetches names/descriptions from `public.translations` using `town\_types\_master.code`.

### 4\. Diff Summary

- Added Columns:

- `is\_active BOOLEAN NOT NULL DEFAULT true`

- `created\_by\_profile\_id UUID REFERENCES public.profiles(id) ON DELETE SET NULL`

- `updated\_by\_profile\_id UUID REFERENCES public.profiles(id) ON DELETE SET NULL`

- Modified Columns:

- None.

- Removed Columns:

- None (original spec `3.3 town\_types\_master.docx` was already aligned with removing `name\_en`, `description\_en`).

- Added DDL Elements:

- `COMMENT ON TABLE` and `COMMENT ON COLUMN` statements.

- Explicit RLS policies for admin, authenticated, and anonymous users, with admin using `public.is\_platform\_admin()` and `WITH CHECK`.

- RLS policies for authenticated/anonymous now filter by `is\_active = true`.

- Modified DDL Elements:

- `updated\_at` trigger function reference standardized to `public.handle\_updated\_at()`.

- Added Documentation/Sections:

- Enhanced details in all 12 sections of the specification.

- "Next-Action Checklist" now includes severity icons and specific references.

- JSON Schema:

- Added `is\_active`, `created\_by\_profile\_id`, `updated\_by\_profile\_id`.

- Updated main description to include version.

- `is\_active` added to `required` properties.