<https://gemini.google.com/u/1/app/bf28f389cd093e55?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/34bd37922031b95a>

<https://gemini.google.com/u/1/app/21b8e85b3e611787>

\* \* \* \* \*

### 3\. Updated Production-Ready Specification

4.10 Establishment Price Ranges Master Table (Version 1.4)

----------------------------------------------------------

This document details the structure, purpose, and considerations for the `establishment\_price\_ranges\_master` table. This table is for categorizing the general price level of commercial establishments like restaurants or cafes. Version 1.4 updates the RLS policies to align with the platform-wide security and authentication strategy using the `public.has\_role()` helper function.

### 1\. Purpose & Primary Use-Cases

The `establishment\_price\_ranges\_master` table defines standardized price range categories (e.g., "Budget Friendly (€)," "Mid-Range (€€)," "Expensive (€€€)") for commercial food and drink establishments. Its purpose is to give pilgrims a quick, visual indication of expected costs, support filtering by price level, and ensure consistent, translatable terminology across the platform.

Key user-story touchpoints:

- Pilgrim: Quickly assessing the general cost of eating at a restaurant or cafe. (Story A4)

- Pilgrim: Filtering food establishments by price range to match their budget.

- Admin/Content Manager: Classifying establishments by a predefined set of price indicators, including managing their active state.

- System/UI: Displaying price range symbols (e.g., "€", "€€", "€€€") and filter options consistently for active price ranges.

### 2\. Schema (Markdown Table)

\*(No change to column structure from Version 1.3)\*

| column | data\_type | constraints | description |

| id | `integer` | Primary Key (Generated as identity always) | Unique identifier for the price range. |

| code | `text` | Unique, Not Null, CHECK (length(code) > 0 AND length(code) &lt;= 50 AND code ~ '^[a-z0-9\_]+$') | Short, stable, machine-readable code (e.g., 'budget', 'mid\_range', 'expensive', 'not\_applicable'). Snake\_case. |

| label | `text` | Not Null, CHECK (length(label) > 0 AND length(label) &lt;= 100) | Human-readable name in the primary reference language (English) for UI display and as a base for translation. Translatable. |

| description | `text` | Nullable | Optional description in the primary reference language (English) providing more context for the price range. Translatable. |

| symbol | `text` | Nullable, CHECK (symbol IS NULL OR length(symbol) &lt;= 10) | Visual symbol for the price range (e.g., "€", "€€", "$"). Max 10 chars. |

| sort\_order | `integer` | Not Null, Default 0 | Determines the display order in UI lists or filters (e.g., cheapest to most expensive). |

| is\_active | `boolean` | Not Null, Default true | True if the price range is active and available for use; false if retired/archived. |

| created\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of record creation. |

| updated\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of last update (auto-updated by trigger). |

| created\_by\_profile\_id | `uuid` | Nullable, Foreign Key to `public.profiles(id)` ON DELETE SET NULL | Profile ID of the user/admin who created this price range record. |

| updated\_by\_profile\_id | `uuid` | Nullable, Foreign Key to `public.profiles(id)` ON DELETE SET NULL | Profile ID of the user/admin who last updated this price range record. |

### 3\. PostgreSQL DDL

\*(DDL for table structure, comments, triggers, and indexes remain the same as Version 1.3. Only the version in the table comment changes.)\*

SQL

```

-- Assumes public.profiles table exists

-- Assumes public.set\_current\_timestamp\_updated\_at() function exists

-- Assumes public.cleanup\_related\_translations(TEXT, TEXT) function and specific per-table wrapper exist

CREATE TABLE public.establishment\_price\_ranges\_master (

id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

code TEXT UNIQUE NOT NULL CHECK (length(code) > 0 AND length(code) <= 50 AND code ~ '^[a-z0-9\_]+$'),

label TEXT NOT NULL CHECK (length(label) > 0 AND length(label) <= 100),

description TEXT NULL,

symbol TEXT NULL CHECK (symbol IS NULL OR length(symbol) <= 10),

sort\_order INTEGER NOT NULL DEFAULT 0,

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 NULL REFERENCES public.profiles(id) ON DELETE SET NULL,

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

);

COMMENT ON TABLE public.establishment\_price\_ranges\_master IS 'Master list of price range categories for commercial establishments. Replaces establishment\_price\_range\_enum. Version 1.4';

-- Column comments from Version 1.3 remain unchanged. E.g.:

COMMENT ON COLUMN public.establishment\_price\_ranges\_master.label IS 'Human-readable name in the primary reference language (English) for UI and as a base for translation. Translatable via the ''translations'' table. Max 100 chars.';

COMMENT ON COLUMN public.establishment\_price\_ranges\_master.is\_active IS 'True if the price range is active and available for use; false if retired/archived. Defaults to true.';

COMMENT ON COLUMN public.establishment\_price\_ranges\_master.created\_by\_profile\_id IS 'Profile ID of the user/admin who created this record.';

COMMENT ON COLUMN public.establishment\_price\_ranges\_master.updated\_by\_profile\_id IS 'Profile ID of the user/admin who last updated this record.';

-- Indexes (including idx\_eprm\_label from previous update)

CREATE INDEX IF NOT EXISTS idx\_eprm\_is\_active ON public.establishment\_price\_ranges\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_eprm\_sort\_order ON public.establishment\_price\_ranges\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_eprm\_label ON public.establishment\_price\_ranges\_master(label);

CREATE INDEX IF NOT EXISTS idx\_eprm\_created\_by\_profile\_id ON public.establishment\_price\_ranges\_master(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_eprm\_updated\_by\_profile\_id ON public.establishment\_price\_ranges\_master(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

-- Trigger for updated\_at

CREATE TRIGGER trigger\_establishment\_price\_ranges\_master\_set\_updated\_at

BEFORE UPDATE ON public.establishment\_price\_ranges\_master

FOR EACH ROW

EXECUTE FUNCTION public.set\_current\_timestamp\_updated\_at();

COMMENT ON TRIGGER trigger\_establishment\_price\_ranges\_master\_set\_updated\_at ON public.establishment\_price\_ranges\_master IS 'Trigger to automatically update updated\_at timestamp on row modification.';

-- Trigger for orphan translation cleanup

CREATE OR REPLACE FUNCTION public.cleanup\_establishment\_price\_ranges\_master\_translations()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'establishment\_price\_ranges\_master'

AND row\_foreign\_key = OLD.id::TEXT;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_establishment\_price\_ranges\_master\_translations

AFTER DELETE ON public.establishment\_price\_ranges\_master

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_establishment\_price\_ranges\_master\_translations();

COMMENT ON TRIGGER trigger\_cleanup\_establishment\_price\_ranges\_master\_translations ON public.establishment\_price\_ranges\_master IS 'Cleans up orphaned translations from public.translations when an establishment\_price\_ranges\_master record is deleted.';

-- Initial Data Example (ensure created\_by\_profile\_id and updated\_by\_profile\_id are set appropriately for seed data)

INSERT INTO public.establishment\_price\_ranges\_master (code, label, symbol, description, sort\_order, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('budget\_friendly', 'Budget Friendly', '€', 'Generally inexpensive, suitable for tight budgets.', 10, true, NULL, NULL),

('mid\_range', 'Mid-Range', '€€', 'Moderately priced, offering good value.', 20, true, NULL, NULL),

('expensive', 'Expensive', '€€€', 'Higher-priced, typically for a more upscale experience.', 30, true, NULL, NULL),

('varies\_check\_menu', 'Varies / Check Menu', NULL, 'Prices vary widely; check menu or inquire locally.', 40, true, NULL, NULL),

('not\_applicable', 'Not Applicable', NULL, 'Price range is not applicable (e.g., for free water sources).', 50, true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "establishment\_price\_range\_master",

"description": "Master list of price range categories for commercial establishments. Version 1.4",

"type": "object",

"properties": {

"id": { /\* ... \*/ },

"code": { /\* ... \*/ },

"label": { /\* ... \*/ },

"description": { /\* ... \*/ },

"symbol": { /\* ... \*/ },

"sort\_order": { /\* ... \*/ },

"is\_active": { /\* ... \*/ },

"created\_at": { /\* ... \*/ },

"updated\_at": { /\* ... \*/ },

"created\_by\_profile\_id": { /\* ... \*/ },

"updated\_by\_profile\_id": { /\* ... \*/ }

},

"required": [ /\* ... \*/ ]

}

```

### 5\. Relationships & Integrity

\*(No change from Version 1.3)\*

- Primary Key: `id` (INTEGER)

- Unique Constraint: `code` must be unique.

- Foreign Key References FROM other tables:

- `food\_water\_sources\_details.establishment\_price\_range\_id` REFERENCES `public.establishment\_price\_ranges\_master(id)` (ON DELETE SET NULL).

- Foreign Key References TO other tables:

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

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

- Data Integrity Notes: Retiring a price range by `is\_active = false`.

### 6\. Multilingual Strategy

\*(No change from Version 1.3)\*

- Translatable Fields: `label`, `description`.

- Translation Management: Via `public.translations` table and orphan cleanup trigger.

### 7\. Role-Based Workflow & RLS Notes

\*(This section is updated to reflect the new auth strategy)\*

- Content Management: This table is typically managed by users with the `admin\_platform` role.

- Lifecycle: Price ranges are made inactive by setting `is\_active = false`. Physical deletion should be rare.

- RLS Policies (Assumes `public.has\_role(TEXT)` helper function exists):

- Public Users (Read-Only on active price ranges):

SQL

```

-- Name: Allow public read access to active price ranges

-- Target: establishment\_price\_ranges\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active price ranges"

ON public.establishment\_price\_ranges\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage price ranges

-- Target: establishment\_price\_ranges\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage price ranges"

ON public.establishment\_price\_ranges\_master FOR ALL

USING (

auth.role() = 'authenticated' AND

public.has\_role('admin\_platform')

) WITH CHECK (

auth.role() = 'authenticated' AND

public.has\_role('admin\_platform')

);

```

- Enable RLS:

SQL

```

ALTER TABLE public.establishment\_price\_ranges\_master ENABLE ROW LEVEL SECURITY;

```

- Notes: RLS must filter by `is\_active = true` for general read access.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- 🟢 Decision: Correctly a lookup table.

- Reasoning: Structured management, translatable names, symbols, sort order, auditability, lifecycle.

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- `label` (translated): For text display.

- `symbol`: For concise visual representation.

- `sort\_order`: Ensures logical listing.

- `description` (translated): For tooltips.

- `is\_active`: UI should only use active ranges.

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- Symbol Usage: Key for UI.

- `not\_applicable` Code: Important for irrelevant cases.

- Subjectivity: `description` can offer guidelines.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3)\*

- Manageable List: Small and stable.

- Flexibility: Easy to add attributes.

- Audit Fields & `is\_active` flag: Robust.

### 12\. Next-Action Checklist

\*(No DDL changes required for this specific update beyond what was in V1.3, the main change is to the RLS policy definition in this document)\*

- 🔴 Verify/Implement RLS Helper: Ensure the `public.has\_role(TEXT)` helper function is correctly implemented and available.

- 🔴 Apply RLS Policies: Implement and thoroughly test the updated RLS policies.

- 🔴 Initial Population/Seed Data: Ensure `created\_by\_profile\_id`/`updated\_by\_profile\_id` are correctly set for seed data.

- 🟢 UI Integration: Plan use of `symbol` and translated `label`.

- 🟢 Translation Entries: Prepare/verify English entries for `label` and `description` in `public.translations`.