<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>

http://gemini.google.com/u/1/app/34bd37922031b95a

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

### 3\. Updated Production-Ready Specification

4.11 Meal Type Tags Master Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `meal\_type\_tags\_master` table, which is part of the food/water module. This table will standardize the tags used to indicate what types of meals an establishment serves. 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 `meal\_type\_tags\_master` table provides a definitive list of meal types that can be associated with food establishments (e.g., "Breakfast," "Lunch," "Dinner," "Snacks"). Its purpose is to enable standardized tagging for consistent filtering, multilingual display of meal types, and potentially associating icons with each meal type.

Key user-story touchpoints:

- Pilgrim: Filtering food establishments based on meals served (e.g., "find places open for dinner"). (Story A4)

- Pilgrim: Quickly seeing what meals an establishment offers (e.g., icons for breakfast, lunch, dinner on a listing).

- Admin/Content Manager: Tagging establishments with the appropriate meal types from a predefined, managed list, and managing the lifecycle of these tags (active/inactive).

- System/UI: Populating filter options for meal types and displaying meal availability in a structured way using active tags.

### 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 meal type tag. |

| 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., 'breakfast\_colazione', 'lunch\_pranzo'). 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 for the meal type in the primary reference language (English). Translatable. |

| icon\_identifier | `text` | Nullable, CHECK (icon\_identifier IS NULL OR length(icon\_identifier) &lt;= 100) | Name, class, or path for a UI icon associated with this meal type. |

| sort\_order | `integer` | Not Null, Default 0 | Determines the display order in UI lists or filters. |

| is\_active | `boolean` | Not Null, Default true | True if the tag is active and available for use; false if retired/archived. Important for array FK validation. |

| 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 meal type tag 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 meal type tag 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.meal\_type\_tags\_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,

icon\_identifier TEXT NULL CHECK (icon\_identifier IS NULL OR length(icon\_identifier) <= 100),

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.meal\_type\_tags\_master IS 'Master list of meal type tags (e.g., breakfast, lunch, dinner). Replaces meal\_type\_tag\_enum. Version 1.4';

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

COMMENT ON COLUMN public.meal\_type\_tags\_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.meal\_type\_tags\_master.is\_active IS 'True if the tag is active and available for use; false if retired. Important for array FK validation.';

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

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

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

CREATE INDEX IF NOT EXISTS idx\_mttm\_is\_active ON public.meal\_type\_tags\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_mttm\_sort\_order ON public.meal\_type\_tags\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_mttm\_label ON public.meal\_type\_tags\_master(label);

CREATE INDEX IF NOT EXISTS idx\_mttm\_created\_by\_profile\_id ON public.meal\_type\_tags\_master(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_mttm\_updated\_by\_profile\_id ON public.meal\_type\_tags\_master(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

-- Trigger for updated\_at

CREATE TRIGGER trigger\_meal\_type\_tags\_master\_set\_updated\_at

BEFORE UPDATE ON public.meal\_type\_tags\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_meal\_type\_tags\_master\_set\_updated\_at ON public.meal\_type\_tags\_master IS 'Trigger to automatically update updated\_at timestamp on row modification.';

-- Trigger for orphan translation cleanup

CREATE OR REPLACE FUNCTION public.cleanup\_meal\_type\_tags\_master\_translations()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'meal\_type\_tags\_master'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_meal\_type\_tags\_master\_translations

AFTER DELETE ON public.meal\_type\_tags\_master

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_meal\_type\_tags\_master\_translations();

COMMENT ON TRIGGER trigger\_cleanup\_meal\_type\_tags\_master\_translations ON public.meal\_type\_tags\_master IS 'Cleans up orphaned translations from public.translations when a meal\_type\_tags\_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.meal\_type\_tags\_master (code, label, icon\_identifier, sort\_order, description, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('breakfast\_colazione', 'Breakfast (Colazione)', 'icon-breakfast', 10, 'Morning meal service.', true, NULL, NULL),

('brunch', 'Brunch', 'icon-brunch', 20, 'Combination of breakfast and lunch, typically served late morning to early afternoon.', true, NULL, NULL),

-- ... other seed data from V1.3 ...

('take\_away\_available', 'Take Away Available', 'icon-take-away', 100, 'Food can be ordered for consumption off-premises.', true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "meal\_type\_tag\_master",

"description": "Master list of meal type tags (e.g., breakfast, lunch, dinner). Version 1.4",

"type": "object",

"properties": {

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

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

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

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

"icon\_identifier": { /\* ... \*/ },

"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.serves\_meal\_type\_tag\_ids` (an `INTEGER[]` column) will contain IDs that reference `meal\_type\_tags\_master.id`.

- 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:

- 🔴 CRITICAL: Database triggers on referencing tables (e.g., `food\_water\_sources\_details`) \*must\* be implemented to ensure all IDs in their `serves\_meal\_type\_tag\_ids` array correspond to valid `id` entries in `meal\_type\_tags\_master` AND that `meal\_type\_tags\_master.is\_active = true`.

- Retiring a meal type tag should be done by setting `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. `regional\_content\_manager` might also manage these if meal types are considered part of regional content characteristics.

- Lifecycle: Meal type tags 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 tags):

SQL

```

-- Name: Allow public read access to active meal type tags

-- Target: meal\_type\_tags\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active meal type tags"

ON public.meal\_type\_tags\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators / Content Managers (Full Control):

SQL

```

-- Name: Allow platform content administrators to manage meal type tags

-- Target: meal\_type\_tags\_master

-- Operation: ALL

-- Role(s): admin\_platform, regional\_content\_manager

CREATE POLICY "Allow platform content administrators to manage meal type tags"

ON public.meal\_type\_tags\_master FOR ALL

USING (

auth.role() = 'authenticated' AND

(public.has\_role('admin\_platform') OR public.has\_role('regional\_content\_manager'))

) WITH CHECK (

auth.role() = 'authenticated' AND

(public.has\_role('admin\_platform') OR public.has\_role('regional\_content\_manager'))

);

```

- Enable RLS:

SQL

```

ALTER TABLE public.meal\_type\_tags\_master ENABLE ROW LEVEL SECURITY;

```

- Notes: RLS must filter by `is\_active = true` for general read access. Array FK validation triggers on referencing tables must also check `is\_active = true`.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- 🟢 Decision: Correctly a lookup table.

- Reasoning: Translatable names, descriptions, icons, sort order, auditability, lifecycle management.

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- `label` (translated): For filter labels, display tags.

- `icon\_identifier`: To show icons.

- `sort\_order`: To list meal types logically.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- Tag Granularity: Initial list is a good start.

- Multiple Selections: Establishments can offer multiple meal types.

- Array FK Integrity: Referencing table's trigger logic must check `is\_active = true`.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3)\*

- Manageable List: Number of distinct meal types likely small.

- Flexibility: Easy to add new meal types or 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.

- 🟠 Array FK Integrity: Ensure validation trigger on `food\_water\_sources\_details` (for `serves\_meal\_type\_tag\_ids`) checks `is\_active = true` in this table.

- 🟢 Icon Set Development: Coordinate with UI/UX team.

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