<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

4.31 Meal Services Master Table (Version 1.5)

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

This document details the structure, purpose, and considerations for the `meal\_services\_master` table. This table will define the standard meal services accommodations can offer. Version 1.5 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\_services\_master` table defines a standardized list of meal services that an accommodation might offer (e.g., "Breakfast Available," "Pilgrim Dinner Option," "Guest Kitchen Access"). Its purpose is to ensure data consistency, support filtering by available meal services, allow for multilingual display of service names, and potentially associate icons with each service for a richer user interface.

Key user-story touchpoints:

- Pilgrim (Anna): Filtering accommodations based on whether they offer specific meals (e.g., breakfast included, dinner available).

- Pilgrim (Anna): Quickly seeing what meal services an accommodation provides using active services.

- Accommodation Host (Marco): Selecting which meal services their B&amp;B offers from a predefined list of active services.

- Admin/Content Manager: Managing the global list of recognized meal services, including their active status.

- System/UI: Populating filter options for meal services and displaying them clearly in accommodation details using active services.

### 2\. Schema (Markdown Table)

\*(No change to column structure from Version 1.4)\*

| column | data\_type | constraints | description |

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

| 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", "DINNER\_PILGRIM"). Uppercase snake\_case. |

| name | `text` | Not Null, CHECK (length(name) > 0 AND length(name) &lt;= 100) | Human-readable name in the primary reference language (English) (e.g., "Breakfast Available"). Translatable via the `translations` table. |

| description | `text` | Nullable | Optional further description of the meal service in the primary reference language (English). Translatable via the `translations` table. |

| 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 service. |

| sort\_order | `integer` | Not Null, Default 0 | Determines the display order of meal services in UI lists or selection interfaces. |

| is\_active | `boolean` | Not Null, Default true | True if the service 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 meal service 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 service record. |

### 3\. PostgreSQL DDL

\*(DDL for table structure, comments, triggers, and indexes remain the same as Version 1.4. 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\_services\_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\_]+$'),

name TEXT NOT NULL CHECK (length(name) > 0 AND length(name) <= 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\_services\_master IS 'Master list of meal services an accommodation might offer (e.g., Breakfast, Pilgrim Dinner). Version 1.5';

-- Column comments from Version 1.4 remain unchanged. E.g.:

COMMENT ON COLUMN public.meal\_services\_master.name IS 'Human-readable name in the primary reference language (English). Translatable via the ''translations'' table. Max 100 chars.';

COMMENT ON COLUMN public.meal\_services\_master.is\_active IS 'True if the service is active and available for use; false if retired. Defaults to true.';

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

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

-- Indexes (including idx\_msm\_name from previous update)

CREATE INDEX IF NOT EXISTS idx\_msm\_is\_active ON public.meal\_services\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_msm\_sort\_order ON public.meal\_services\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_msm\_name ON public.meal\_services\_master(name);

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

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

-- Trigger for updated\_at

CREATE TRIGGER trigger\_meal\_services\_master\_set\_updated\_at

BEFORE UPDATE ON public.meal\_services\_master

FOR EACH ROW

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

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

-- Trigger for orphan translation cleanup

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

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'meal\_services\_master'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_meal\_services\_master\_translations

AFTER DELETE ON public.meal\_services\_master

FOR EACH ROW

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

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

('BREAKFAST\_AVAILABLE', 'Breakfast Available', 'icon-breakfast', 10, 'Breakfast service is offered, may be included or at an additional cost.', true, NULL, NULL),

('LUNCH\_PACKED\_AVAILABLE', 'Packed Lunch Available', 'icon-packed-lunch', 20, 'Facility to provide a packed lunch for pilgrims, usually for a fee.', true, NULL, NULL),

-- ... other seed data from V1.4 ...

('VENDING\_MACHINE\_FOOD\_DRINK', 'Vending Machine (Food/Drink)', 'icon-vending-food', 70, 'On-site vending machines for snacks or beverages.', true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.4)\*

JSON

```

{

"title": "meal\_service\_master",

"description": "Master list of meal services an accommodation might offer. Version 1.5",

"type": "object",

"properties": {

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

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

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

"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 1.4)\*

- Primary Key: `id` (INTEGER)

- Unique Constraint: `code` must be unique.

- Foreign Key References FROM other tables:

- `accommodation\_meal\_services.meal\_service\_id` REFERENCES `public.meal\_services\_master(id)` (ON DELETE RESTRICT recommended on junction).

- 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 service by `is\_active = false`.

### 6\. Multilingual Strategy

\*(No change from Version 1.4)\*

- Translatable Fields: `name`, `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: Meal services are made inactive by setting `is\_active = false`. Physical deletion is restricted by FK from `accommodation\_meal\_services` if a service is in use.

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

- Public Users (Read-Only on active services):

SQL

```

-- Name: Allow public read access to active meal services master list

-- Target: meal\_services\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active meal services master list"

ON public.meal\_services\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage meal services master list

-- Target: meal\_services\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage meal services master list"

ON public.meal\_services\_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.meal\_services\_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 1.4)\*

- 🟢 Decision: Correctly a lookup table.

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

### 9\. UI/UX Enablement

\*(No change from Version 1.4)\*

- `name` (translated): For display in filters and listings.

- `icon\_identifier`: For icons.

- `description` (translated): For tooltips.

- `sort\_order`: Ensures logical listing.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.4)\*

- `code`: Must be unique, stable (UPPER\_SNAKE\_CASE).

- Clarity: Names/descriptions must be clear (e.g., "Breakfast Included" vs. "Breakfast Available").

- Iconography: Consistent icons needed.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.4)\*

- Lookup Table Structure: Scalable.

- Manageable List: Number of service types expected to be moderate.

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

### 12\. Next-Action Checklist

\*(No DDL changes required for this specific update beyond what was in V1.4, 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.

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

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