<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

4.3 Accommodation Types Master Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `accommodation\_types\_master` table. 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 `accommodation\_types\_master` table serves as a centralized, definitive list of all valid accommodation types available on the platform (e.g., "Pilgrim Hostel," "B&amp;B," "Hotel"). Its purpose is to ensure data consistency, provide standardized categories for filtering and display, and support multilingual representation of accommodation types with associated icons and descriptions.

Key user-story touchpoints:

- Pilgrim: Filtering accommodation searches by type. (Story A3)

- Pilgrim: Understanding the type of an accommodation from its listing via a clear, translated name and potentially an icon.

- Accommodation Host: Selecting the appropriate type for their listing during setup or update. (Story B1)

- Admin/Content Manager: Managing the official list of recognized accommodation types, ensuring clarity, appropriate categorization, and lifecycle (active/inactive).

- System/UI: Driving UI elements like filter dropdowns, icons next to accommodation listings, and grouping search results based on active types.

### 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 accommodation type. |

| 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., 'ospitale\_pilgrim\_hostel', 'b\_and\_b'). 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 of the accommodation 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 accommodation type. |

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

| is\_active | `boolean` | Not Null, Default true | True if the category 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 accommodation type 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 accommodation type 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.accommodation\_types\_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.accommodation\_types\_master IS 'Master list of official accommodation types (e.g., hostel, B&B, hotel). Replaces accommodation\_type\_enum. Version 1.4';

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

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

COMMENT ON COLUMN public.accommodation\_types\_master.is\_active IS 'True if the category is active and available for use; false if retired/archived. Defaults to true.';

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

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

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

CREATE INDEX IF NOT EXISTS idx\_atm\_is\_active ON public.accommodation\_types\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_atm\_sort\_order ON public.accommodation\_types\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_atm\_label ON public.accommodation\_types\_master(label);

CREATE INDEX IF NOT EXISTS idx\_atm\_created\_by\_profile\_id ON public.accommodation\_types\_master(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_atm\_updated\_by\_profile\_id ON public.accommodation\_types\_master(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

-- Trigger for updated\_at

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

BEFORE UPDATE ON public.accommodation\_types\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_accommodation\_types\_master\_set\_updated\_at ON public.accommodation\_types\_master IS 'Trigger to automatically update updated\_at timestamp on row modification.';

-- Trigger for orphan translation cleanup

CREATE OR REPLACE FUNCTION public.cleanup\_accommodation\_types\_master\_translations()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

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

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_accommodation\_types\_master\_translations

AFTER DELETE ON public.accommodation\_types\_master

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_accommodation\_types\_master\_translations();

COMMENT ON TRIGGER trigger\_cleanup\_accommodation\_types\_master\_translations ON public.accommodation\_types\_master IS 'Cleans up orphaned translations from public.translations when an accommodation\_types\_master record is deleted.';

-- Seed data (as per Version 1.3, with audit columns populated appropriately)

INSERT INTO public.accommodation\_types\_master (code, label, description, icon\_identifier, sort\_order, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('ospitale\_pilgrim\_hostel', 'Pilgrim Hostel (Ospitale)', 'Simple lodging specifically for pilgrims, often with shared facilities.', 'icon-hostel', 10, true, NULL, NULL),

('b\_and\_b', 'Bed & Breakfast', 'Accommodation in a private home offering breakfast.', 'icon-b-and-b', 20, true, NULL, NULL),

-- ... other seed data ...

('other', 'Other Accommodation', 'Other types of lodging not fitting standard categories.', 'icon-other-pin', 900, true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "accommodation\_type\_master",

"description": "Master list of official accommodation types (e.g., hostel, B&B, hotel). Replaces accommodation\_type\_enum. 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:

- `accommodations.accommodation\_type\_id` REFERENCES `public.accommodation\_types\_master(id)` (ON DELETE RESTRICT).

- 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 an accommodation type 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. Depending on operational needs, `regional\_content\_manager` might also be granted some management capabilities, though typically master data like this is centrally managed.

- Lifecycle: Accommodation types are made inactive by setting `is\_active = false`. Physical deletion is restricted by foreign key constraints if the type is in use and should be an exceptional administrative action by an `admin\_platform`.

- RLS Policies (Assumes `public.has\_role(TEXT)` helper function exists and checks `public.profiles.roles` for the current `auth.uid()`):

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

SQL

```

-- Name: Allow public read access to active accommodation types

-- Target: accommodation\_types\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active accommodation types"

ON public.accommodation\_types\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage accommodation types

-- Target: accommodation\_types\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage accommodation types"

ON public.accommodation\_types\_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.accommodation\_types\_master ENABLE ROW LEVEL SECURITY;

```

- Notes:

- The `public.has\_role('admin\_platform')` function is crucial. If other roles (e.g., `regional\_content\_manager`) need similar access, the `USING` and `WITH CHECK` conditions should be expanded (e.g., `(public.has\_role('admin\_platform') OR public.has\_role('regional\_content\_manager'))`). For core master data, restricting full CRUD to `admin\_platform` is often preferred.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- 🟢 Decision: Correctly a lookup table.

- Reasoning: Richness, i18n, maintainability, data integrity, auditability, lifecycle.

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

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

- `icon\_identifier`: For icons.

- `sort\_order`: For consistent presentation.

- `description` (translated): For tooltips.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- `code` vs. `label`: Stable system ID vs. display name.

- Icon Strategy: UI assets match `icon\_identifier`.

- Data Management: Administrative task; lifecycle via `is\_active`.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3)\*

- Lookup Table Structure: Scalable, flexible.

- Audit Fields & `is\_active` flag: Good for history and lifecycle.

### 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 (e.g., system admin UUID).

- 🟢 Review Role Assignments: Confirm which roles (`admin\_platform`, `regional\_content\_manager`, etc.) should have management rights over this specific master table and adjust RLS if necessary.