<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

4.28 Room Types Master Table (Version 1.5)

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

This document details the structure, purpose, and considerations for the `room\_types\_master` table. This table is essential for classifying the different types of rooms 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 `room\_types\_master` table defines a standardized list of room types available in accommodations (e.g., "Private Double Room with Ensuite," "4-Bed Mixed Dormitory," "Apartment Unit"). Its purpose is to ensure data consistency for room classifications, support detailed filtering by room type, allow for multilingual display of room type names, and provide essential characteristics of each room type (like typical capacity and style) for both pilgrim information and host data entry.

Key user-story touchpoints:

- Pilgrim (Anna): Filtering accommodations based on desired room types (e.g., private room vs. dormitory bed). (Story A3)

- Pilgrim (Anna): Understanding the specifics of room types offered by an accommodation.

- Accommodation Host (Marco): Specifying the types and quantities of rooms their B&amp;B offers, linking to these standard definitions. (Story B1)

- Admin/Content Manager: Managing the global list of recognized room types, including their active status, to ensure clarity and comprehensiveness.

- System/UI: Populating filter options for room types and displaying room type information in a structured way on accommodation listings using active room types.

### 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 room type. |

| room\_type\_code | `text` | Unique, Not Null, CHECK (length(room\_type\_code) > 0 AND length(room\_type\_code) &lt;= 50 AND room\_type\_code ~ '^[A-Z0-9\_]+$') | Short, stable, machine-readable code (e.g., "PRIVATE\_DOUBLE\_ENSUITE"). Uppercase snake\_case. |

| name | `text` | Not Null, CHECK (length(name) > 0 AND length(name) &lt;= 150) | Human-readable name in the primary reference language (English). Translatable via the `translations` table. |

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

| typical\_capacity\_persons | `smallint` | Not Null, CHECK (typical\_capacity\_persons > 0 AND typical\_capacity\_persons &lt; 100) | Typical number of people this room type accommodates. |

| is\_dorm\_style | `boolean` | Not Null, Default false | True if this typically involves a bed in a shared dormitory-style room. |

| is\_private\_occupancy | `boolean` | Not Null, Default true | True if this room type implies private occupancy for the booking party (not typically shared with strangers if booked as such). False for individual dorm beds. |

| has\_private\_bathroom\_option | `boolean` | Not Null, Default false | Indicates if this room type \*can\* come with a private (ensuite) bathroom. Actual presence confirmed at `accommodation\_room\_configurations`. |

| is\_entire\_unit\_rental | `boolean` | Not Null, Default false | True if this represents renting an entire self-contained unit like an apartment. |

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

| is\_active | `boolean` | Not Null, Default true | True if the room type 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 room 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 room type 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.room\_types\_master (

id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

room\_type\_code TEXT UNIQUE NOT NULL CHECK (length(room\_type\_code) > 0 AND length(room\_type\_code) <= 50 AND room\_type\_code ~ '^[A-Z0-9\_]+$'),

name TEXT NOT NULL CHECK (length(name) > 0 AND length(name) <= 150),

description TEXT NULL,

typical\_capacity\_persons SMALLINT NOT NULL CHECK (typical\_capacity\_persons > 0 AND typical\_capacity\_persons < 100),

is\_dorm\_style BOOLEAN NOT NULL DEFAULT FALSE,

is\_private\_occupancy BOOLEAN NOT NULL DEFAULT TRUE,

has\_private\_bathroom\_option BOOLEAN NOT NULL DEFAULT FALSE,

is\_entire\_unit\_rental BOOLEAN NOT NULL DEFAULT FALSE,

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.room\_types\_master IS 'Master list defining standard types of rooms available in accommodations. Version 1.5';

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

COMMENT ON COLUMN public.room\_types\_master.name IS 'Human-readable name in the primary reference language (English). Translatable via the ''translations'' table. Max 150 chars.';

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

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

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

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

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

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

CREATE INDEX IF NOT EXISTS idx\_rtm\_name ON public.room\_types\_master(name);

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

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

-- Trigger for updated\_at

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

BEFORE UPDATE ON public.room\_types\_master

FOR EACH ROW

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

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

-- Trigger for orphan translation cleanup

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

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

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

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

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

AFTER DELETE ON public.room\_types\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_cleanup\_room\_types\_master\_translations ON public.room\_types\_master IS 'Cleans up orphaned translations from public.translations when a room\_types\_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.room\_types\_master (room\_type\_code, name, typical\_capacity\_persons, is\_dorm\_style, is\_private\_occupancy, has\_private\_bathroom\_option, is\_entire\_unit\_rental, sort\_order, description, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('DORM\_BED\_MIXED', 'Bed in Mixed Dormitory', 1, true, false, false, false, 10, 'A single bed in a dormitory room shared with other guests of any gender. Bathroom facilities are typically shared.', true, NULL, NULL),

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

('OTHER\_ROOM\_TYPE', 'Other Room Type', 1, false, true, false, false, 900, 'A type of room not covered by other standard classifications.', true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.4)\*

JSON

```

{

"title": "room\_type\_master",

"description": "Master list defining standard types of rooms available in accommodations. Version 1.5",

"type": "object",

"properties": {

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

"room\_type\_code": { /\* ... \*/ },

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

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

"typical\_capacity\_persons": { /\* ... \*/ },

"is\_dorm\_style": { /\* ... \*/ },

"is\_private\_occupancy": { /\* ... \*/ },

"has\_private\_bathroom\_option": { /\* ... \*/ },

"is\_entire\_unit\_rental": { /\* ... \*/ },

"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: `room\_type\_code` must be unique.

- Foreign Key References FROM other tables:

- `accommodation\_room\_configurations.room\_type\_id` REFERENCES `public.room\_types\_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 room type 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: Room types are made inactive by setting `is\_active = false`. Physical deletion is restricted by FK from `accommodation\_room\_configurations` if a room type is in use.

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

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

SQL

```

-- Name: Allow public read access to active room types master list

-- Target: room\_types\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active room types master list"

ON public.room\_types\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage room types master list

-- Target: room\_types\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage room types master list"

ON public.room\_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.room\_types\_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: Room types have several defining characteristics. Lookup table allows central management, translatable names/descriptions, auditability, lifecycle.

### 9\. UI/UX Enablement

\*(No change from Version 1.4)\*

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

- `description` (translated): For tooltips.

- Boolean flags: For quick characteristic display.

- `typical\_capacity\_persons`: Informs users and aids capacity calculation.

- `sort\_order`: Ensures logical listing.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.4)\*

- `room\_type\_code`: Stable, unique (UPPER\_SNAKE\_CASE).

- Clarity of Definitions: Boolean flags and descriptions must be clear.

- `has\_private\_bathroom\_option`: Indicates possibility; actual configuration in `accommodation\_room\_configurations`.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.4)\*

- Manageable List: Number of distinct room types will be moderate.

- Flexibility: Easy to add new room types or refine attributes.

- 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.

- 🟢 Review Definitions: Ensure boolean flags and descriptions clearly define each room type.

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