<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

4.4 Booking Statuses Master Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `booking\_statuses\_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 `booking\_statuses\_master` table provides a centralized, standard list of possible booking availability statuses for accommodations (e.g., "Open - Bookings Available," "Likely Full," "Closed Seasonally"). Its purpose is to ensure consistent terminology for booking statuses, support clear communication of availability to pilgrims, and enable multilingual display of these statuses.

Key user-story touchpoints:

- Pilgrim: Quickly understanding the likely booking availability of an accommodation. (Story A3, B2)

- Accommodation Host: Updating their current booking availability status for pilgrims to see. (Story B2)

- Admin/Content Manager: Managing the set of official booking statuses, including their active state.

- System/UI: Displaying availability statuses with appropriate labels (and potentially colors/icons) in accommodation listings or summaries, based on active statuses.

### 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 booking status. |

| 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., 'open\_bookings\_available', 'full'). 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 booking status in the primary reference language (English). Translatable. |

| is\_positive\_status | `boolean` | Not Null, Default true | Indicates if the status generally represents positive availability (e.g., 'open' is true, 'full' is false). For UI styling. |

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

| is\_active | `boolean` | Not Null, Default true | True if the status 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 booking status 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 booking status 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.booking\_statuses\_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,

is\_positive\_status BOOLEAN NOT NULL DEFAULT TRUE,

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.booking\_statuses\_master IS 'Master list of booking availability statuses for accommodations. Replaces booking\_status\_enum. Version 1.4';

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

COMMENT ON COLUMN public.booking\_statuses\_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.booking\_statuses\_master.is\_active IS 'True if the status is active and available for use; false if retired/archived. Defaults to true.';

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

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

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

CREATE INDEX IF NOT EXISTS idx\_bsm\_is\_active ON public.booking\_statuses\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_bsm\_sort\_order ON public.booking\_statuses\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_bsm\_label ON public.booking\_statuses\_master(label);

CREATE INDEX IF NOT EXISTS idx\_bsm\_created\_by\_profile\_id ON public.booking\_statuses\_master(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_bsm\_updated\_by\_profile\_id ON public.booking\_statuses\_master(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

-- Trigger for updated\_at

CREATE TRIGGER trigger\_booking\_statuses\_master\_set\_updated\_at

BEFORE UPDATE ON public.booking\_statuses\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_booking\_statuses\_master\_set\_updated\_at ON public.booking\_statuses\_master IS 'Trigger to automatically update updated\_at timestamp on row modification.';

-- Trigger for orphan translation cleanup

CREATE OR REPLACE FUNCTION public.cleanup\_booking\_statuses\_master\_translations()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'booking\_statuses\_master'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_booking\_statuses\_master\_translations

AFTER DELETE ON public.booking\_statuses\_master

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_booking\_statuses\_master\_translations();

COMMENT ON TRIGGER trigger\_cleanup\_booking\_statuses\_master\_translations ON public.booking\_statuses\_master IS 'Cleans up orphaned translations from public.translations when a booking\_statuses\_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.booking\_statuses\_master (code, label, description, is\_positive\_status, sort\_order, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('unknown', 'Unknown', 'Availability has not been specified or recently updated.', false, 0, true, NULL, NULL),

('open\_bookings\_available', 'Open - Bookings Available', 'Likely has space and is accepting bookings.', true, 10, true, NULL, NULL),

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

('closed\_seasonally', 'Closed Seasonally', 'Closed for the season (e.g., winter closure).', false, 80, true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "booking\_status\_master",

"description": "Master list of booking availability statuses for accommodations. Version 1.4",

"type": "object",

"properties": {

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

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

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

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

"is\_positive\_status": { /\* ... \*/ },

"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.booking\_availability\_status\_id` REFERENCES `public.booking\_statuses\_master(id)` (ON DELETE SET NULL, with a `DEFAULT` in `accommodations`).

- 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 status 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: Booking statuses are made inactive by setting `is\_active = false`. Physical deletion is generally not recommended if a status has been used, to preserve historical meaning, but `ON DELETE SET NULL` (with a default to 'unknown') on the referencing `accommodations` table handles cases where deletion might occur.

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

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

SQL

```

-- Name: Allow public read access to active booking statuses

-- Target: booking\_statuses\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active booking statuses"

ON public.booking\_statuses\_master FOR SELECT

USING (is\_active = true);

```

- Accommodation Hosts (Read-Only on active statuses, for selection in their UIs): The public policy above suffices.

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage booking statuses

-- Target: booking\_statuses\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage booking statuses"

ON public.booking\_statuses\_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.booking\_statuses\_master ENABLE ROW LEVEL SECURITY;

```

- Notes: RLS must filter by `is\_active = true` for general read access. The `DEFAULT` value for `accommodations.booking\_availability\_status\_id` ensures that if a status is hard-deleted (despite `ON DELETE SET NULL`), new records in `accommodations` don't fail, though retiring via `is\_active = false` is preferred.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- 🟢 Decision: Correctly a lookup table.

- Reasoning: Translatable fields, metadata (`is\_positive\_status`, `sort\_order`), maintainability, auditability, lifecycle.

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- `label` (translated): For filter labels and display names.

- `description` (translated): For tooltips.

- `is\_positive\_status`: For UI visual cues.

- `sort\_order`: For logical presentation.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- `code`: Must be unique, stable, especially for the `DEFAULT` value in `accommodations`.

- Default Status: 'unknown' status crucial for `accommodations` default.

- Host Workflow: Hosts select these statuses.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3)\*

- Lookup Table Structure: Scalable.

- Adding Attributes: Easy.

### 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.

- 🟢 Default Value in `accommodations`: Verify default correctly references the 'unknown' status ID.

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