<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

4.29 Accommodation Room Configurations Table (Version 1.5)

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

This document details the structure, purpose, and considerations for the `accommodation\_room\_configurations` table. This table is crucial for detailing the specific room offerings of an accommodation, including quantities and pricing. Version 1.5 updates the RLS policies to align with the platform-wide security and authentication strategy using the `public.has\_role()` helper function and checks for active master data.

### 1\. Purpose & Primary Use-Cases

The `accommodation\_room\_configurations` table links an accommodation to the specific types of rooms it offers (from `room\_types\_master`), specifying the quantity of each room type available and their pricing details. This allows for a detailed inventory of an accommodation's capacity and supports precise information for pilgrims looking for particular room setups.

Key user-story touchpoints:

- Pilgrim (Anna): Seeing the exact types of rooms available at an accommodation (e.g., "2 x Private Twin Ensuite," "4 x Dorm Beds"), their prices, and any specific notes for those rooms (Story A3).

- Pilgrim (Anna): Filtering based on the availability of specific room types and understanding per-person vs. per-room pricing (Story A3).

- Accommodation Host (Marco): Managing their detailed room inventory, including the number of each room type, pricing, and notes specific to those configurations (e.g., "ground floor, accessible," "bunk beds") (Story B1).

- System/UI: Calculating total capacity accurately (for the `accommodations\_capacity\_summary\_view`), displaying available room configurations with prices, and enabling precise filtering based on room availability.

### 2\. Schema (Markdown Table)

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

| column | data\_type | constraints | description |

| id | `bigint` | Primary Key (Generated as identity always) | Unique identifier for this specific room configuration entry. |

| accommodation\_waypoint\_id | `bigint` | Not Null, Foreign Key to `accommodations(waypoint\_id)` ON DELETE CASCADE, Part of UNIQUE constraint `uq\_accommodation\_room\_type` | Links to the specific accommodation this room configuration belongs to. |

| room\_type\_id | `integer` | Not Null, Foreign Key to `room\_types\_master(id)` ON DELETE RESTRICT, Part of UNIQUE constraint `uq\_accommodation\_room\_type` | Links to the type of room being configured (e.g., "Private Double Ensuite"). RESTRICT prevents deleting a master room type if in use. |

| count\_of\_this\_room\_type | `smallint` | Not Null, CHECK (`count\_of\_this\_room\_type` > 0 AND `count\_of\_this\_room\_type` &lt; 1000) | How many rooms of this specific type are available (e.g., "2" of "PRIVATE\_DOUBLE\_ENSUITE"). |

| price\_amount | `numeric(10, 2)` | Nullable, CHECK (`price\_amount` IS NULL OR `price\_amount` >= 0) | Price for one unit of this room type at this accommodation. Numeric for precision. |

| price\_currency\_code | `char(3)` | Not Null, Default 'EUR', CHECK (length(`price\_currency\_code`) = 3 AND `price\_currency\_code` = upper(`price\_currency\_code`)) | ISO 4217 currency code for `price\_amount` (e.g., "EUR"). V1 defaults to EUR. |

| price\_is\_per\_person | `boolean` | Not Null, Default false | Indicates if `price\_amount` is per person (e.g., for dorm beds) or per room/unit. |

| price\_notes | `text` | Nullable | Specific pricing notes for this room type (e.g., "Weekend surcharge applies," "Breakfast optional for €X extra"). Primary reference language (English) text. Translatable. |

| room\_specific\_notes | `text` | Nullable | Other notes specific to this room configuration at this accommodation (e.g., "Ground floor, accessible," "Max 2 adults + 1 child"). Primary reference language (English) text. Translatable. |

| created\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of record creation. |

| created\_by\_profile\_id | `uuid` | Nullable, Foreign Key to `public.profiles(id)` ON DELETE SET NULL | Profile ID of the user who created this room configuration. |

| updated\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of last update (auto-updated by trigger). |

| updated\_by\_profile\_id | `uuid` | Nullable, Foreign Key to `public.profiles(id)` ON DELETE SET NULL | Profile ID of the user who last updated this room configuration. |

### 3\. PostgreSQL DDL

\*(DDL for table structure, comments, triggers, and existing indexes remain the same as Version 1.4. Only the version in the table comment changes.)\*

SQL

```

-- Assumes public.accommodations, public.room\_types\_master, public.profiles tables exist

-- 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\_room\_configurations (

id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

accommodation\_waypoint\_id BIGINT NOT NULL,

room\_type\_id INTEGER NOT NULL,

count\_of\_this\_room\_type SMALLINT NOT NULL CHECK (count\_of\_this\_room\_type > 0 AND count\_of\_this\_room\_type < 1000),

price\_amount NUMERIC(10, 2) NULL CHECK (price\_amount IS NULL OR price\_amount >= 0),

price\_currency\_code CHAR(3) NOT NULL DEFAULT 'EUR' CHECK (length(price\_currency\_code) = 3 AND price\_currency\_code = upper(price\_currency\_code)),

price\_is\_per\_person BOOLEAN NOT NULL DEFAULT FALSE,

price\_notes TEXT NULL,

room\_specific\_notes TEXT NULL,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

created\_by\_profile\_id UUID NULL,

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

updated\_by\_profile\_id UUID NULL,

CONSTRAINT fk\_accommodation\_waypoint

FOREIGN KEY(accommodation\_waypoint\_id)

REFERENCES public.accommodations(waypoint\_id)

ON DELETE CASCADE,

CONSTRAINT fk\_room\_type

FOREIGN KEY(room\_type\_id)

REFERENCES public.room\_types\_master(id)

ON DELETE RESTRICT,

CONSTRAINT fk\_created\_by\_profile

FOREIGN KEY(created\_by\_profile\_id)

REFERENCES public.profiles(id)

ON DELETE SET NULL,

CONSTRAINT fk\_updated\_by\_profile

FOREIGN KEY(updated\_by\_profile\_id)

REFERENCES public.profiles(id)

ON DELETE SET NULL,

CONSTRAINT uq\_accommodation\_room\_type UNIQUE (accommodation\_waypoint\_id, room\_type\_id)

);

COMMENT ON TABLE public.accommodation\_room\_configurations IS 'Details the specific room types, quantities, and pricing available at a particular accommodation. Version 1.5';

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

COMMENT ON COLUMN public.accommodation\_room\_configurations.price\_notes IS 'Specific pricing notes for this room type (e.g., "Weekend surcharge"). Primary reference language (English) text. Translatable via ''translations'' table using this record''s `id` as row\_foreign\_key.';

COMMENT ON COLUMN public.accommodation\_room\_configurations.room\_specific\_notes IS 'Other notes specific to this room configuration (e.g., "Ground floor, accessible"). Primary reference language (English) text. Translatable via ''translations'' table using this record''s `id` as row\_foreign\_key.';

COMMENT ON COLUMN public.accommodation\_room\_configurations.created\_by\_profile\_id IS 'Profile ID of the user who created this configuration.';

COMMENT ON COLUMN public.accommodation\_room\_configurations.updated\_by\_profile\_id IS 'Profile ID of the user who last updated this configuration.';

-- Trigger for updated\_at

CREATE TRIGGER trigger\_accommodation\_room\_configurations\_set\_updated\_at

BEFORE UPDATE ON public.accommodation\_room\_configurations

FOR EACH ROW

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

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

-- Trigger for orphan translation cleanup

CREATE OR REPLACE FUNCTION public.cleanup\_accommodation\_room\_configurations\_translations()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'accommodation\_room\_configurations'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_accommodation\_room\_configurations\_translations

AFTER DELETE ON public.accommodation\_room\_configurations

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_accommodation\_room\_configurations\_translations();

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

-- Indexes (including audit FK indexes from previous update)

CREATE INDEX idx\_arc\_accommodation\_waypoint\_id ON public.accommodation\_room\_configurations(accommodation\_waypoint\_id);

CREATE INDEX idx\_arc\_room\_type\_id ON public.accommodation\_room\_configurations(room\_type\_id);

CREATE INDEX IF NOT EXISTS idx\_arc\_created\_by\_profile\_id ON public.accommodation\_room\_configurations(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_arc\_updated\_by\_profile\_id ON public.accommodation\_room\_configurations(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.4)\*

JSON

```

{

"title": "accommodation\_room\_configuration",

"description": "Details the specific room types, quantities, and pricing available at a particular accommodation. Version 1.5",

"type": "object",

"properties": { /\* ... all properties as in Version 1.4 ... \*/ },

"required": [ /\* ... as in Version 1.4 ... \*/ ],

"primary\_key": ["id"],

"uniqueKeys": [ /\* ... as in Version 1.4 ... \*/ ]

}

```

### 5\. Relationships & Integrity

\*(No change from Version 1.4)\*

- Primary Key: `id` (BIGINT, surrogate key).

- Unique Constraint: `uq\_accommodation\_room\_type` on (`accommodation\_waypoint\_id`, `room\_type\_id`).

- Foreign Keys: As defined in Version 1.4.

- Mermaid ER Snippet: (As provided in Version 1.4 documentation).

### 6\. Multilingual Strategy

\*(No change from Version 1.4)\*

- `price\_notes` and `room\_specific\_notes` are translatable.

- Translation Management: Via `public.translations` table (using `id` as `row\_foreign\_key`) and orphan cleanup trigger.

### 7\. Role-Based Workflow & RLS Notes

\*(This section is updated to reflect the new auth strategy)\*

- Key Fields for Workflow: `count\_of\_this\_room\_type`, `price\_amount`, `price\_notes`, `room\_specific\_notes` are managed by hosts or platform content team roles. Audit fields track changes.

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

- Public Users (Read-Only for configurations of published accommodations):

SQL

```

-- Name: Allow public read access to room configurations of published accommodations

-- Target: accommodation\_room\_configurations

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to room configurations of published accommodations"

ON public.accommodation\_room\_configurations FOR SELECT

USING (

EXISTS (

SELECT 1 FROM public.accommodations acc

JOIN public.waypoints w ON acc.waypoint\_id = w.id

WHERE acc.waypoint\_id = accommodation\_room\_configurations.accommodation\_waypoint\_id

AND w.deleted\_at IS NULL

AND w.content\_visibility\_status\_id = (SELECT cs.id FROM public.content\_statuses\_master cs WHERE cs.code = 'published' LIMIT 1)

AND (

SELECT rtm.is\_active FROM public.room\_types\_master rtm

WHERE rtm.id = accommodation\_room\_configurations.room\_type\_id

) = true -- Ensure linked room type is active

)

);

```

- Accommodation Hosts (Manage configurations for their own accommodations):

SQL

```

-- Name: Hosts can manage room configurations for their own accommodations

-- Target: accommodation\_room\_configurations

-- Operation: ALL

-- Role(s): authenticated (checked via host\_profile\_id on parent accommodation)

CREATE POLICY "Hosts can manage room configurations for their own accommodations"

ON public.accommodation\_room\_configurations FOR ALL

USING (

auth.role() = 'authenticated' AND

EXISTS (

SELECT 1 FROM public.accommodations acc

WHERE acc.waypoint\_id = accommodation\_room\_configurations.accommodation\_waypoint\_id

AND acc.host\_profile\_id = auth.uid()

)

) WITH CHECK (

auth.role() = 'authenticated' AND

EXISTS (

SELECT 1 FROM public.accommodations acc

WHERE acc.waypoint\_id = accommodation\_room\_configurations.accommodation\_waypoint\_id

AND acc.host\_profile\_id = auth.uid()

)

AND (TG\_OP = 'DELETE' OR accommodation\_waypoint\_id = NEW.accommodation\_waypoint\_id) -- Prevent reparenting

AND (TG\_OP = 'DELETE' OR TG\_OP = 'UPDATE' OR ( -- For INSERT, check the linked room type is active

SELECT rtm.is\_active FROM public.room\_types\_master rtm WHERE rtm.id = NEW.room\_type\_id

) = true)

);

```

- Platform Content Team (Admins/Managers - Full control over links):

SQL

```

-- Name: Platform content team can manage all accommodation room configurations

-- Target: accommodation\_room\_configurations

-- Operation: ALL

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

CREATE POLICY "Platform content team can manage all accommodation room configurations"

ON public.accommodation\_room\_configurations FOR ALL

USING (

auth.role() = 'authenticated' AND

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

) WITH CHECK (

auth.role() = 'authenticated' AND

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

AND (TG\_OP = 'DELETE' OR TG\_OP = 'UPDATE' OR ( -- For INSERT, check the linked room type is active

SELECT rtm.is\_active FROM public.room\_types\_master rtm WHERE rtm.id = NEW.room\_type\_id

) = true)

);

```

- Enable RLS:

SQL

```

ALTER TABLE public.accommodation\_room\_configurations ENABLE ROW LEVEL SECURITY;

```

- Notes:

- Public read policy now also checks if the linked `room\_types\_master.is\_active = true`.

- `INSERT` and `UPDATE` policies for hosts and admins now include a `WITH CHECK` condition to ensure links are only made to `active` room types.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.4)\*

- Not applicable directly. Links to `room\_types\_master` (a lookup table).

### 9\. UI/UX Enablement

\*(No change from Version 1.4)\*

- Detailed Room Breakdown: Fundamental for showing room types, quantities, prices.

- Accurate Capacity Calculation: Drives `accommodations\_capacity\_summary\_view`.

- Pricing Clarity: `price\_amount`, `price\_currency\_code`, `price\_is\_per\_person`.

- `room\_specific\_notes` (translated): For important details.

### 10\. Key Considerations & Definitions

\*(No change from Version 1.4)\*

- Currency Handling: `price\_currency\_code` is `NOT NULL DEFAULT 'EUR'`.

- Total Capacity Calculation: Definitive source for `accommodations\_capacity\_summary\_view`.

- Unique Constraint: Enforces one configuration per room type per accommodation.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.4, existing "Future Consideration" point remains relevant)\*

- Normalized Structure: Scalable.

- Pricing Flexibility: Separate fields allow for future multi-currency.

- Parent Table Timestamp Updates (V2+): Consider if CUD operations here should also trigger an update to `accommodations.updated\_at`.

### 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, including the checks against `room\_types\_master.is\_active`.

- 🟢 Verify Total Capacity Logic: Ensure `accommodations\_capacity\_summary\_view` correctly uses this table.

- 🟢 Verify Currency Implementation: Confirm `price\_currency\_code` default and constraint.

- 🟢 Translation Entries: Prepare/verify English entries for `price\_notes` and `room\_specific\_notes` in `public.translations`.