<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

4.27 Accommodation Amenities Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `accommodation\_amenities` table, which links accommodations to the amenities they offer. 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\_amenities` table serves as a junction (or linking) table that creates a many-to-many relationship between accommodations and the amenities they offer. It allows each accommodation to be associated with multiple amenities from the `amenities\_master` table, and for each amenity to be offered by multiple accommodations. It can also store context-specific notes about an amenity at a particular accommodation.

Key user-story touchpoints:

- Pilgrim (Anna): Viewing the specific list of amenities available at a chosen accommodation, along with any relevant notes (e.g., "Wi-Fi (charges apply)"). (Story A3)

- Pilgrim (Anna): Filtering accommodations based on the presence of one or more specific amenities.

- Accommodation Host (Marco): Selecting and managing the list of amenities their B&amp;B offers, and adding specific notes for some of them. (Story B1)

- System/UI: Querying this table to dynamically construct the list of amenities for display on an accommodation's detail page.

### 2\. Schema (Markdown Table)

\*(No change to column structure from Version 1.3)\*

| column | data\_type | constraints | description |

| id | `uuid` | Primary Key (Default: `gen\_random\_uuid()`) | Unique identifier for this specific accommodation-amenity link. |

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

| amenity\_id | `integer` | Not Null, Foreign Key to `amenities\_master(id)` ON DELETE RESTRICT, Part of UNIQUE constraint `uq\_accommodation\_amenity` | Links to the specific amenity from the master list. RESTRICT prevents deleting a master amenity if in use. |

| notes\_on\_amenity | `text` | Nullable | Specific notes about this amenity at this particular accommodation (e.g., "WiFi only in common areas, €2/day"). Primary reference language (English) text. Translatable. |

| created\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of when this specific amenity was linked to the accommodation. |

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

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

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

### 3\. PostgreSQL DDL

\*(DDL for table structure, comments, triggers, and existing indexes remain the same as Version 1.3. Only the version in the table comment changes.)\*

SQL

```

-- Assumes public.accommodations, public.amenities\_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\_amenities (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

accommodation\_waypoint\_id BIGINT NOT NULL,

amenity\_id INTEGER NOT NULL,

notes\_on\_amenity TEXT NULL,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

created\_by\_profile\_id UUID NULL,

updated\_by\_profile\_id UUID NULL,

CONSTRAINT uq\_accommodation\_amenity UNIQUE (accommodation\_waypoint\_id, amenity\_id),

CONSTRAINT fk\_accommodation\_waypoint

FOREIGN KEY(accommodation\_waypoint\_id)

REFERENCES public.accommodations(waypoint\_id)

ON DELETE CASCADE,

CONSTRAINT fk\_amenity

FOREIGN KEY(amenity\_id)

REFERENCES public.amenities\_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

);

COMMENT ON TABLE public.accommodation\_amenities IS 'Junction table linking accommodations to the amenities they offer, with optional context-specific notes. Version 1.4';

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

COMMENT ON COLUMN public.accommodation\_amenities.notes\_on\_amenity IS 'Specific notes about this amenity at this accommodation (e.g., "WiFi only in common areas, €2/day"). Primary reference language (English) text. Translatable via the ''translations'' table using this record''s `id` as row\_foreign\_key.';

COMMENT ON COLUMN public.accommodation\_amenities.created\_by\_profile\_id IS 'Profile ID of the user who linked this amenity to the accommodation.';

COMMENT ON COLUMN public.accommodation\_amenities.updated\_by\_profile\_id IS 'Profile ID of the user who last updated the notes for this linked amenity.';

-- Trigger for updated\_at

CREATE TRIGGER trigger\_accommodation\_amenities\_set\_updated\_at

BEFORE UPDATE ON public.accommodation\_amenities

FOR EACH ROW

WHEN (OLD.notes\_on\_amenity IS DISTINCT FROM NEW.notes\_on\_amenity)

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

COMMENT ON TRIGGER trigger\_accommodation\_amenities\_set\_updated\_at ON public.accommodation\_amenities IS 'Trigger to automatically update updated\_at timestamp when notes\_on\_amenity is modified.';

-- Trigger for orphan translation cleanup

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

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'accommodation\_amenities'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_accommodation\_amenities\_translations

AFTER DELETE ON public.accommodation\_amenities

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_accommodation\_amenities\_translations();

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

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

CREATE INDEX idx\_accommodation\_amenities\_amenity\_id ON public.accommodation\_amenities(amenity\_id);

COMMENT ON INDEX public.idx\_accommodation\_amenities\_amenity\_id IS 'Index to efficiently find all accommodations offering a specific amenity.';

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

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

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "accommodation\_amenity\_link",

"description": "Links an accommodation to a specific amenity and allows for context-specific notes. Version 1.4",

"type": "object",

"properties": { /\* ... all properties as in Version 1.3 ... \*/ },

"required": [ /\* ... as in Version 1.3 ... \*/ ],

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

"uniqueKeys": [ /\* ... as in Version 1.3 ... \*/ ]

}

```

### 5\. Relationships & Integrity

\*(No change from Version 1.3)\*

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

- Unique Constraint: `uq\_accommodation\_amenity` on (`accommodation\_waypoint\_id`, `amenity\_id`).

- Foreign Keys: As defined in Version 1.3.

- Mermaid ER Snippet: (As provided in Version 1.3 documentation).

### 6\. Multilingual Strategy

\*(No change from Version 1.3)\*

- `notes\_on\_amenity`: 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: `notes\_on\_amenity` can be updated by hosts or by platform content team roles who have rights to the parent accommodation. Audit fields track changes to these specific links.

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

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

SQL

```

-- Name: Allow public read access to amenities of published accommodations

-- Target: accommodation\_amenities

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to amenities of published accommodations"

ON public.accommodation\_amenities 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\_amenities.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 am.is\_active FROM public.amenities\_master am

WHERE am.id = accommodation\_amenities.amenity\_id

) = true -- Ensure linked amenity is active

)

);

```

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

SQL

```

-- Name: Hosts can manage amenities for their own accommodations

-- Target: accommodation\_amenities

-- Operation: ALL

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

CREATE POLICY "Hosts can manage amenities for their own accommodations"

ON public.accommodation\_amenities FOR ALL

USING (

auth.role() = 'authenticated' AND

EXISTS (

SELECT 1 FROM public.accommodations acc

WHERE acc.waypoint\_id = accommodation\_amenities.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\_amenities.accommodation\_waypoint\_id

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

)

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

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

SELECT am.is\_active FROM public.amenities\_master am WHERE am.id = NEW.amenity\_id

) = true)

);

```

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

SQL

```

-- Name: Platform content team can manage all accommodation amenities

-- Target: accommodation\_amenities

-- Operation: ALL

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

CREATE POLICY "Platform content team can manage all accommodation amenities"

ON public.accommodation\_amenities 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 amenity is active

SELECT am.is\_active FROM public.amenities\_master am WHERE am.id = NEW.amenity\_id

) = true)

);

```

- Enable RLS:

SQL

```

ALTER TABLE public.accommodation\_amenities ENABLE ROW LEVEL SECURITY;

```

- Notes:

- The public read policy now also checks if the linked `amenities\_master.is\_active = true`.

- The `INSERT` and `UPDATE` policies for hosts and admins now include a `WITH CHECK` condition to ensure that only links to `active` amenities can be created or maintained (unless it's a DELETE operation).

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- Not applicable directly to this table. Links to `amenities\_master` (a lookup table).

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- Enables UI to list amenities with context-specific notes.

- `amenity\_id` allows joining to `amenities\_master` for rich display.

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- Unique Constraint: `(accommodation\_waypoint\_id, amenity\_id)` is unique.

- `ON DELETE` Behavior: `CASCADE` from `accommodations`, `RESTRICT` to `amenities\_master`.

- `notes\_on\_amenity` Mutability: `updated\_at` trigger is conditional.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3, existing "Future Consideration" point remains relevant)\*

- Junction Table Standard: Scalable M-M pattern.

- Performance: Indexes support common queries. Audit FK indexes added.

- Parent Table Timestamp Updates (V2+): Consider if significant changes 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 V1.3, 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, including the checks against `amenities\_master.is\_active`.

- 🟢 Verify Other Aspects: Confirm all other aspects from Version 1.3 (triggers, translation strategy) are correctly implemented and tested.