<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

4.30 Accommodation Payment Methods Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `accommodation\_payment\_methods` table, which links accommodations to the payment methods they accept. Version 1.4 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\_payment\_methods` table serves as a junction (linking) table to establish a many-to-many relationship between accommodations and the payment methods they accept (defined in `payment\_methods\_master`). It allows each accommodation to list multiple accepted payment methods and provides a space for specific notes regarding a payment method at that particular location.

Key user-story touchpoints:

- Pilgrim (Anna): Seeing a clear list of accepted payment methods (e.g., "Cash," "Visa," "PayPal") for an accommodation, potentially with icons and specific notes like "Visa accepted for amounts over €20." (Story A3)

- Pilgrim (Anna): Filtering accommodations based on whether they accept specific payment types (e.g., credit cards).

- Accommodation Host (Marco): Selecting which payment methods their establishment accepts and adding any relevant conditions or notes for each. (Story B1)

- System/UI: Dynamically querying and displaying the accepted payment methods for each accommodation.

### 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-payment method link. |

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

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

| notes\_on\_method | `text` | Nullable | Specific notes for this payment method at this particular accommodation. Primary reference language (English) text. Translatable. |

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

| updated\_at | `timestamp with time zone` | Not Null, Default `now()` | Timestamp of when `notes\_on\_method` 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 payment method. |

| 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 payment method. |

### 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.payment\_methods\_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\_payment\_methods (

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

accommodation\_waypoint\_id BIGINT NOT NULL,

payment\_method\_id INTEGER NOT NULL,

notes\_on\_method 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\_payment\_method UNIQUE (accommodation\_waypoint\_id, payment\_method\_id),

CONSTRAINT fk\_accommodation\_waypoint

FOREIGN KEY(accommodation\_waypoint\_id)

REFERENCES public.accommodations(waypoint\_id)

ON DELETE CASCADE,

CONSTRAINT fk\_payment\_method

FOREIGN KEY(payment\_method\_id)

REFERENCES public.payment\_methods\_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\_payment\_methods IS 'Junction table linking accommodations to the payment methods they accept, with optional context-specific notes. Version 1.4';

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

COMMENT ON COLUMN public.accommodation\_payment\_methods.notes\_on\_method IS 'Specific notes for this payment method at this accommodation. Primary reference language (English) text. Translatable via the ''translations'' table using this record''s `id` as row\_foreign\_key.';

COMMENT ON COLUMN public.accommodation\_payment\_methods.created\_by\_profile\_id IS 'Profile ID of the user who linked this payment method.';

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

-- Trigger for updated\_at

CREATE TRIGGER trigger\_accommodation\_payment\_methods\_set\_updated\_at

BEFORE UPDATE ON public.accommodation\_payment\_methods

FOR EACH ROW

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

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

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

-- Trigger for orphan translation cleanup

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

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'accommodation\_payment\_methods'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER trigger\_cleanup\_accommodation\_payment\_methods\_translations

AFTER DELETE ON public.accommodation\_payment\_methods

FOR EACH ROW

EXECUTE FUNCTION public.cleanup\_accommodation\_payment\_methods\_translations();

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

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

CREATE INDEX idx\_acc\_payment\_methods\_payment\_method\_id ON public.accommodation\_payment\_methods(payment\_method\_id);

COMMENT ON INDEX public.idx\_acc\_payment\_methods\_payment\_method\_id IS 'Index to efficiently find all accommodations accepting a specific payment method.';

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

CREATE INDEX IF NOT EXISTS idx\_apm\_updated\_by\_profile\_id ON public.accommodation\_payment\_methods(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\_payment\_method\_link",

"description": "Links an accommodation to a specific payment method they accept 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\_payment\_method` on (`accommodation\_waypoint\_id`, `payment\_method\_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\_method`: 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\_method` can be updated by hosts or by platform content team roles who have rights to the parent accommodation. Audit fields track these specific link changes.

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

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

SQL

```

-- Name: Allow public read access to payment methods of published accommodations

-- Target: accommodation\_payment\_methods

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to payment methods of published accommodations"

ON public.accommodation\_payment\_methods 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\_payment\_methods.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 pmm.is\_active FROM public.payment\_methods\_master pmm

WHERE pmm.id = accommodation\_payment\_methods.payment\_method\_id

) = true -- Ensure linked payment method is active

)

);

```

- Accommodation Hosts (Manage payment methods for their own accommodations):

SQL

```

-- Name: Hosts can manage payment methods for their own accommodations

-- Target: accommodation\_payment\_methods

-- Operation: ALL

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

CREATE POLICY "Hosts can manage payment methods for their own accommodations"

ON public.accommodation\_payment\_methods FOR ALL

USING (

auth.role() = 'authenticated' AND

EXISTS (

SELECT 1 FROM public.accommodations acc

WHERE acc.waypoint\_id = accommodation\_payment\_methods.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\_payment\_methods.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 payment method is active

SELECT pmm.is\_active FROM public.payment\_methods\_master pmm WHERE pmm.id = NEW.payment\_method\_id

) = true)

);

```

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

SQL

```

-- Name: Platform content team can manage all accommodation payment methods

-- Target: accommodation\_payment\_methods

-- Operation: ALL

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

CREATE POLICY "Platform content team can manage all accommodation payment methods"

ON public.accommodation\_payment\_methods 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 payment method is active

SELECT pmm.is\_active FROM public.payment\_methods\_master pmm WHERE pmm.id = NEW.payment\_method\_id

) = true)

);

```

- Enable RLS:

SQL

```

ALTER TABLE public.accommodation\_payment\_methods ENABLE ROW LEVEL SECURITY;

```

- Notes:

- Public read policy now also checks if the linked `payment\_methods\_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` payment methods.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

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

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- Enables UI to list accepted payment methods with context-specific notes.

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- `ON DELETE RESTRICT` for `payment\_method\_id`: Important.

- Translation of `notes\_on\_method`: Using surrogate `id` for `row\_foreign\_key`.

### 11\. Scalability & Future-Proofing

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

- Standard Junction Table: Scalable M-M pattern.

- Flexibility: `notes\_on\_method` covers most V1 cases.

- 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 V1.3, 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 `payment\_methods\_master.is\_active`.

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