<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

4.14 Payment Methods Master Table (Version 1.4)

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

This document details the structure, purpose, and considerations for the `payment\_methods\_master` table, which is used by both the accommodations module and is planned for use by `food\_water\_sources\_details`. 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 `payment\_methods\_master` table provides a standardized list of accepted payment methods (e.g., "Cash (EUR)," "Visa," "Mastercard," "PayPal," "Satispay"). Its purpose is to allow accommodations and other commercial waypoints (like restaurants or shops) to clearly indicate which payment options they support, enabling pilgrims to plan accordingly and ensuring consistent, translatable terminology and associated icons.

Key user-story touchpoints:

- Pilgrim: Knowing in advance which payment methods are accepted at an accommodation or restaurant, especially in areas where cash might be preferred or card acceptance is limited. (Story A3, A4)

- Pilgrim: Filtering or quickly identifying establishments that accept their preferred payment methods.

- Accommodation Host/Establishment Owner: Specifying the payment methods they accept for their listing. (Story B1)

- Admin/Content Manager: Managing the global list of recognized payment methods, including their active state.

- System/UI: Displaying payment method icons and information consistently, and enabling related filtering using active methods.

### 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 payment method. |

| 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., "CASH\_EUR", "VISA"). Uppercase 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 in the primary reference language (English) providing more context about the payment method. Translatable. |

| icon\_identifier | `text` | Nullable, CHECK (icon\_identifier IS NULL OR length(icon\_identifier) &lt;= 100) | Name, class, or path for a UI icon associated with this payment method. |

| category | `text` | Nullable, CHECK (category IS NULL OR length(category) &lt;= 50) | Broad category for grouping in the primary reference language (English) (e.g., "Cash", "Card", "Digital Wallet"). Translatable. |

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

| is\_active | `boolean` | Not Null, Default true | True if the method is active and available for use; false if retired/archived. Important for array FK validation. |

| 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 payment method 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 payment method 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.payment\_methods\_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,

icon\_identifier TEXT NULL CHECK (icon\_identifier IS NULL OR length(icon\_identifier) <= 100),

category TEXT NULL CHECK (category IS NULL OR length(category) <= 50),

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.payment\_methods\_master IS 'Master list of accepted payment methods (e.g., Cash, Visa, PayPal). Version 1.4';

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

COMMENT ON COLUMN public.payment\_methods\_master.label IS 'Human-readable name in the primary reference language (English) for UI and as a base for translation. Translatable via the ''translations'' table. Max 100 chars.';

COMMENT ON COLUMN public.payment\_methods\_master.category IS 'Broad category for grouping in the primary reference language (English) (e.g., "Card", "Digital Wallet"). Translatable via the ''translations'' table. Max 50 chars.';

COMMENT ON COLUMN public.payment\_methods\_master.is\_active IS 'True if the method is active and available for use; false if retired. Important for array FK validation.';

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

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

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

CREATE INDEX IF NOT EXISTS idx\_pmm\_is\_active ON public.payment\_methods\_master(is\_active);

CREATE INDEX IF NOT EXISTS idx\_pmm\_sort\_order ON public.payment\_methods\_master(sort\_order);

CREATE INDEX IF NOT EXISTS idx\_pmm\_category ON public.payment\_methods\_master(category) WHERE category IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_pmm\_label ON public.payment\_methods\_master(label);

CREATE INDEX IF NOT EXISTS idx\_pmm\_created\_by\_profile\_id ON public.payment\_methods\_master(created\_by\_profile\_id) WHERE created\_by\_profile\_id IS NOT NULL;

CREATE INDEX IF NOT EXISTS idx\_pmm\_updated\_by\_profile\_id ON public.payment\_methods\_master(updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

-- Trigger for updated\_at

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

BEFORE UPDATE ON public.payment\_methods\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_payment\_methods\_master\_set\_updated\_at ON public.payment\_methods\_master IS 'Trigger to automatically update updated\_at timestamp on row modification.';

-- Trigger for orphan translation cleanup

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

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'DELETE' THEN

DELETE FROM public.translations

WHERE table\_identifier = 'payment\_methods\_master'

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

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

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

AFTER DELETE ON public.payment\_methods\_master

FOR EACH ROW

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

COMMENT ON TRIGGER trigger\_cleanup\_payment\_methods\_master\_translations ON public.payment\_methods\_master IS 'Cleans up orphaned translations from public.translations when a payment\_methods\_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.payment\_methods\_master (code, label, icon\_identifier, category, sort\_order, description, is\_active, created\_by\_profile\_id, updated\_by\_profile\_id) VALUES

('CASH\_EUR', 'Cash (EUR)', 'icon-cash-eur', 'Cash', 10, 'Payment in Euro currency notes and coins.', true, NULL, NULL),

('VISA', 'Visa', 'icon-visa', 'Card', 20, 'Visa credit or debit cards accepted.', true, NULL, NULL),

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

('OTHER\_DIGITAL', 'Other Digital Payment', 'icon-other-digital', 'Digital Wallet', 120, 'Accepts other types of digital or mobile payments.', true, NULL, NULL);

```

### 4\. JSON Schema Mirror

\*(No change from Version 1.3)\*

JSON

```

{

"title": "payment\_method\_master",

"description": "Master list of accepted payment methods. Version 1.4",

"type": "object",

"properties": {

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

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

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

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

"icon\_identifier": { /\* ... \*/ },

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

"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:

- `accommodation\_payment\_methods.payment\_method\_id` REFERENCES `public.payment\_methods\_master(id)` (ON DELETE RESTRICT).

- `food\_water\_sources\_details.payment\_method\_tag\_ids` (an `INTEGER[]` column) will contain IDs that reference `payment\_methods\_master.id`.

- 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:

- 🔴 CRITICAL: Array FK validation triggers on referencing tables (like `food\_water\_sources\_details`) must check `payment\_methods\_master.is\_active = true`.

- Retiring a payment method should be done by setting `is\_active = false`.

### 6\. Multilingual Strategy

\*(No change from Version 1.3)\*

- Translatable Fields: `label`, `description`, `category`.

- 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: Payment methods are made inactive by setting `is\_active = false`. Physical deletion should be rare and is restricted if in use by `accommodation\_payment\_methods`.

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

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

SQL

```

-- Name: Allow public read access to active payment methods

-- Target: payment\_methods\_master

-- Operation: SELECT

-- Role(s): anon, authenticated

CREATE POLICY "Allow public read access to active payment methods"

ON public.payment\_methods\_master FOR SELECT

USING (is\_active = true);

```

- Platform Administrators (Full Control):

SQL

```

-- Name: Allow platform administrators to manage payment methods

-- Target: payment\_methods\_master

-- Operation: ALL

-- Role(s): admin\_platform

CREATE POLICY "Allow platform administrators to manage payment methods"

ON public.payment\_methods\_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.payment\_methods\_master ENABLE ROW LEVEL SECURITY;

```

- Notes: RLS must filter by `is\_active = true` for general read access. Array FK validation on other tables must also check `is\_active = true`.

### 8\. ENUM vs Lookup Discussion

\*(No change from Version 1.3)\*

- 🟢 Decision: Correctly a lookup table.

- Reasoning: Translatable fields, icon association, categorization, auditability, lifecycle.

### 9\. UI/UX Enablement

\*(No change from Version 1.3)\*

- `label` (translated): For displaying accepted payment methods.

- `icon\_identifier`: To show logos/icons.

- `category` (translated): For grouping.

- `sort\_order`: To list common methods first.

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

### 10\. Key Considerations & Definitions

\*(No change from Version 1.3)\*

- Global vs. Regional Methods: Include common international and relevant regional ones.

- Icon Sourcing: Needed for `icon\_identifier`.

- Specificity: E.g., `CASH\_EUR`.

- Array FK Integrity: Referencing table's trigger logic must check `is\_active = true`.

### 11\. Scalability & Future-Proofing

\*(No change from Version 1.3)\*

- Manageable List: Number of common payment methods is manageable.

- Flexibility: Easy to add new payment methods. `category` aids grouping.

- Audit Fields & `is\_active` flag: Robust.

### 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.

- 🟠 Array FK Integrity: Ensure validation trigger on `food\_water\_sources\_details` (for `payment\_method\_tag\_ids`) checks `is\_active = true` in this table.

- 🟢 Icon Set Development: Coordinate with UI/UX team.

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