<https://gemini.google.com/u/1/app/ea09d2cc0b5505fe?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/fd47ede3d905a3f2

<https://gemini.google.com/u/1/app/440838222ac514d7>

Production-Ready Specification: public.profiles Table

Version: 2.3 (Checklist V2-05-17-B Applied, Role Sync & Last Activity Added)

Date: May 18, 2025

1. Purpose & Primary Use-Cases

Mission Statement: To store application-specific user data, extending Supabase's built-in `auth.users` table via a one-to-one relationship. This table manages user identity for display, assigns roles for permissions (which are then synchronized to `auth.users.raw\_app\_meta\_data.roles` for JWT inclusion), stores preferences for personalization, contributor information, tracks account status, last user activity, and audit trails for administrative updates.

Key User-Story Touchpoints:

• Pilgrim (Anna): Underpins identity, preferences (A6: language via `preferred\_language\_code`, A2/A3: units via `preferred\_units\_of\_measure`), personalization (experience level, interests), and links to her contributions (A8: votes, tips - though these are in separate linking tables). Roles assigned here are available in her JWT. Her `last\_activity\_at` timestamp helps the platform provide relevant recent content or notifications.

• Host (Marco): Defines his role (via `roles` array, synchronized to JWT), stores listing-related contact details (`contact\_public\_email`, `contributor\_organization\_name`), and manages his account status (`account\_status`) for platform access (Story B1).

• Regional Manager (Sofia): Defines her role (via `roles` array, synchronized to JWT) and identity for content management tasks (Stories C1, C2).

• Platform Administrator: Central table for user management (Story D1), role assignment (via `roles` array, synchronized to JWT), status monitoring (`account\_status`), and observing user activity trends via `last\_activity\_at`.

2. Schema

| column | data\_type | constraints | description |

| :------------------------------ | :------------------------ | :---------------------------------------------------------------------------------------------------------------------------------------- | :--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- |

| id | uuid | Primary Key, References `auth.users(id)` ON DELETE CASCADE | User's ID from `auth.users` table. Forms the 1:1 link. Profile deleted if auth user is deleted. |

| roles | text[] | Not Null | Array of role codes (e.g., `{"pilgrim\_user", "accommodation\_host"}`). Values validated by `check\_profile\_roles` trigger. Synchronized to `auth.users.raw\_app\_meta\_data.roles`. |

| username | text | Nullable, Unique (`uq\_profiles\_username`) | Optional unique username for community features. Unique if provided. |

| full\_name | text | Nullable | User's full name for internal or formal use. |

| public\_display\_name | text | Nullable, Unique (`uq\_profiles\_public\_display\_name`) | Name shown publicly (e.g., on tips). User-set or derived. Unique if provided. |

| public\_avatar\_media\_id | uuid | Nullable, References `media(id)` ON DELETE SET NULL | 🟠 ID of the media item (from `media` table) to be used as the public avatar. Allows leveraging the central media system. |

| public\_bio | text | Nullable | Short public biography (e.g., max 500 characters, enforced by application). User-generated content. |

| preferred\_language\_code | text | Not Null, Default 'en', References `languages\_master(language\_code)` ON UPDATE CASCADE ON DELETE RESTRICT, CHECK (`preferred\_language\_code` ~ '^[a-z]{2}(-[A-Z]{2})?$') | 🟠 User's preferred language (ISO 639-1 or IETF tag, e.g., 'en', 'it', 'en-US'). FK to `languages\_master`. Pattern allows for regional codes. Default 'en'. |

| preferred\_units\_of\_measure | units\_preference\_enum | Not Null, Default 'metric' | User's preference for distances/elevations ('metric', 'imperial'). |

| preferred\_timezone | text | Nullable, Default 'Europe/Rome' | User's preferred IANA timezone string (e.g., 'America/New\_York'). |

| pilgrim\_experience\_level | pilgrim\_experience\_enum | Nullable | For 'pilgrim\_user' role, self-assessed experience level. |

| pilgrimage\_interests\_tags | text[] | Nullable | For pilgrim users, array of interest tags (e.g., `{"spirituality", "history"}`). |

| contributor\_organization\_name | text | Nullable | Organization name if user contributes on its behalf. |

| contributor\_organization\_role | text | Nullable | User's role within that organization. |

| contact\_public\_email | text | Nullable, CHECK (`contact\_public\_email` IS NULL OR `contact\_public\_email` ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$') | 🟠 Publicly visible email address, if provided and distinct from login. Validated format. |

| website\_url\_profile | text | Nullable, CHECK (`website\_url\_profile` IS NULL OR `website\_url\_profile` ~\* '^https?://.+') | 🟠 Personal or organizational website URL. Validated format. |

| notification\_preferences\_json | jsonb | Nullable | User preferences for platform notifications (e.g., `{"new\_trail\_warnings": true}`). Structure defined by application. |

| is\_profile\_publicly\_visible | boolean | Not Null, Default false | Master switch for public visibility of `public\_\*` profile fields. |

| contribution\_score | integer | Not Null, Default 0, CHECK (`contribution\_score` >= 0) | Score based on positive contributions. |

| account\_status | user\_account\_status\_enum | Not Null, Default 'active' | Current status of the user's application account. Initial status may be set by `handle\_new\_user` trigger. |

| terms\_accepted\_at | timestamp with time zone | Nullable | Timestamp of last terms and conditions acceptance. |

| last\_login\_at | timestamp with time zone | Nullable | Timestamp of user's last login (updated by application). |

| last\_activity\_at | timestamp with time zone | Nullable | Timestamp of the last recorded user activity (e.g., interaction, content submission). Mechanism for updates (API, triggers) to be defined. |

| created\_at | timestamp with time zone | Not Null, Default `now()` | Timestamp of profile record creation (`created\_by` is implicitly `profiles.id`). |

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

| updated\_by\_profile\_id | uuid | Nullable, References `public.profiles(id)` ON DELETE SET NULL | Profile ID of another user (typically an admin) who last made significant updates to this profile record. ON DELETE SET NULL. |

3. PostgreSQL DDL

-- ENUM Types (ensure these are created globally first as per module build order)

DO $$ BEGIN

IF NOT EXISTS (SELECT 1 FROM pg\_type WHERE typname = 'units\_preference\_enum') THEN CREATE TYPE public.units\_preference\_enum AS ENUM ('metric', 'imperial'); END IF;

IF NOT EXISTS (SELECT 1 FROM pg\_type WHERE typname = 'pilgrim\_experience\_enum') THEN CREATE TYPE public.pilgrim\_experience\_enum AS ENUM ('novice\_first\_pilgrimage', 'intermediate\_few\_pilgrimages', 'experienced\_many\_pilgrimages', 'long\_distance\_veteran'); END IF;

IF NOT EXISTS (SELECT 1 FROM pg\_type WHERE typname = 'user\_account\_status\_enum') THEN CREATE TYPE public.user\_account\_status\_enum AS ENUM ('active', 'pending\_verification', 'email\_unconfirmed', 'suspended\_by\_admin', 'deactivated\_by\_user'); END IF;

END$$;

-- Table Definition

CREATE TABLE public.profiles (

id uuid NOT NULL PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,

roles text[] NOT NULL,

username text NULL UNIQUE,

full\_name text NULL,

public\_display\_name text NULL UNIQUE,

public\_avatar\_media\_id uuid NULL REFERENCES public.media(id) ON DELETE SET NULL,

public\_bio text NULL,

preferred\_language\_code text NOT NULL DEFAULT 'en' REFERENCES public.languages\_master(language\_code) ON UPDATE CASCADE ON DELETE RESTRICT,

preferred\_units\_of\_measure public.units\_preference\_enum NOT NULL DEFAULT 'metric',

preferred\_timezone text NULL DEFAULT 'Europe/Rome',

pilgrim\_experience\_level public.pilgrim\_experience\_enum NULL,

pilgrimage\_interests\_tags text[] NULL,

contributor\_organization\_name text NULL,

contributor\_organization\_role text NULL,

contact\_public\_email text NULL,

website\_url\_profile text NULL,

notification\_preferences\_json jsonb NULL,

is\_profile\_publicly\_visible boolean NOT NULL DEFAULT false,

contribution\_score integer NOT NULL DEFAULT 0,

account\_status public.user\_account\_status\_enum NOT NULL DEFAULT 'active',

terms\_accepted\_at timestamp with time zone NULL,

last\_login\_at timestamp with time zone NULL,

last\_activity\_at timestamp with time zone NULL, -- New column

created\_at timestamp with time zone NOT NULL DEFAULT now(),

updated\_at timestamp with time zone NOT NULL DEFAULT now(),

updated\_by\_profile\_id uuid NULL REFERENCES public.profiles(id) ON DELETE SET NULL,

CONSTRAINT check\_preferred\_language\_code\_format CHECK (preferred\_language\_code ~ '^[a-z]{2}(-[A-Z]{2})?$'),

CONSTRAINT check\_contact\_public\_email\_format CHECK (contact\_public\_email IS NULL OR contact\_public\_email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),

CONSTRAINT check\_website\_url\_profile\_format CHECK (website\_url\_profile IS NULL OR website\_url\_profile ~\* '^https?://.+'),

CONSTRAINT check\_contribution\_score\_non\_negative CHECK (contribution\_score >= 0)

);

-- Indexes

CREATE INDEX idx\_profiles\_roles ON public.profiles USING GIN (roles);

CREATE INDEX idx\_profiles\_pilgrimage\_interests\_tags ON public.profiles USING GIN (pilgrimage\_interests\_tags);

CREATE INDEX idx\_profiles\_account\_status ON public.profiles (account\_status);

CREATE INDEX idx\_profiles\_updated\_by\_profile\_id ON public.profiles (updated\_by\_profile\_id) WHERE updated\_by\_profile\_id IS NOT NULL;

CREATE INDEX idx\_profiles\_last\_activity\_at ON public.profiles (last\_activity\_at DESC NULLS LAST) WHERE last\_activity\_at IS NOT NULL; -- Index for new column

-- Comments

COMMENT ON TABLE public.profiles IS 'Stores application-specific user data, extending Supabase Auth users. Manages identity, roles (synchronized to auth.users), preferences, contributor context, account status, activity timestamps, and admin update audit. Version 2.3.';

COMMENT ON COLUMN public.profiles.id IS 'User''s ID from auth.users table (PK, FK). Profile deleted if auth user is deleted. Implicitly `created\_by`.';

COMMENT ON COLUMN public.profiles.roles IS 'Array of role codes (e.g., {"pilgrim\_user"}). Validated by `check\_profile\_roles` trigger. Synchronized to `auth.users.raw\_app\_meta\_data.roles`.';

COMMENT ON COLUMN public.profiles.last\_activity\_at IS 'Timestamp of the last recorded significant user activity (e.g., interaction, content submission). Mechanism for updates (API calls, specific DB triggers) to be defined.';

COMMENT ON COLUMN public.profiles.updated\_by\_profile\_id IS 'Profile ID of another user (typically an admin) who last made significant updates to this profile record. ON DELETE SET NULL.';

4. Triggers/Functions

-- Trigger for general updated\_at timestamp

CREATE TRIGGER handle\_profiles\_updated\_at

BEFORE UPDATE ON public.profiles

FOR EACH ROW

EXECUTE FUNCTION extensions.moddatetime('updated\_at');

-- Trigger function for checking roles array integrity

CREATE OR REPLACE FUNCTION public.check\_profile\_roles()

RETURNS TRIGGER AS $$

DECLARE role\_code\_val TEXT; invalid\_roles TEXT[]; active\_role\_exists BOOLEAN;

BEGIN

IF TG\_OP = 'INSERT' OR NEW.roles IS DISTINCT FROM OLD.roles THEN

IF NEW.roles IS NULL OR array\_length(NEW.roles, 1) IS NULL OR array\_length(NEW.roles, 1) = 0 THEN RAISE EXCEPTION 'profiles.roles cannot be empty or NULL.'; END IF;

FOREACH role\_code\_val IN ARRAY NEW.roles LOOP

SELECT EXISTS (SELECT 1 FROM public.user\_roles\_master urm WHERE urm.role\_code = role\_code\_val AND urm.is\_role\_active = TRUE AND urm.deleted\_at IS NULL) INTO active\_role\_exists;

IF NOT active\_role\_exists THEN invalid\_roles := array\_append(invalid\_roles, role\_code\_val); END IF;

END LOOP;

IF array\_length(invalid\_roles, 1) > 0 THEN RAISE EXCEPTION 'Invalid, inactive, or deleted role(s) in profiles.roles: %', array\_to\_string(invalid\_roles, ', '); END IF;

END IF; RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

COMMENT ON FUNCTION public.check\_profile\_roles() IS 'Ensures that roles assigned in public.profiles.roles exist, are active, and not soft-deleted in public.user\_roles\_master. Runs as SECURITY DEFINER.';

-- Apply the roles check trigger

CREATE TRIGGER trigger\_check\_profile\_roles\_on\_insert\_or\_update

BEFORE INSERT OR UPDATE OF roles ON public.profiles

FOR EACH ROW

EXECUTE FUNCTION public.check\_profile\_roles();

COMMENT ON TRIGGER trigger\_check\_profile\_roles\_on\_insert\_or\_update ON public.profiles IS 'Validates roles against user\_roles\_master before profile insert or roles update.';

-- Trigger function to synchronize profiles.roles to auth.users.raw\_app\_meta\_data.roles

CREATE OR REPLACE FUNCTION public.sync\_profile\_roles\_to\_auth\_user()

RETURNS TRIGGER AS $$

BEGIN

UPDATE auth.users

SET raw\_app\_meta\_data = COALESCE(raw\_app\_meta\_data, '{}'::jsonb) || jsonb\_build\_object('roles', NEW.roles)

WHERE id = NEW.id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

COMMENT ON FUNCTION public.sync\_profile\_roles\_to\_auth\_user() IS 'Synchronizes the roles from public.profiles to auth.users.raw\_app\_meta\_data.roles for JWT inclusion. Runs as SECURITY DEFINER. Ensure appropriate search\_path if function becomes complex.';

-- Apply the roles synchronization trigger

CREATE TRIGGER trigger\_sync\_roles\_to\_auth\_user\_on\_profile\_change

AFTER INSERT OR UPDATE OF roles ON public.profiles

FOR EACH ROW

EXECUTE FUNCTION public.sync\_profile\_roles\_to\_auth\_user();

COMMENT ON TRIGGER trigger\_sync\_roles\_to\_auth\_user\_on\_profile\_change ON public.profiles IS 'When roles in public.profiles are inserted or updated, synchronizes them to auth.users.raw\_app\_meta\_data.roles.';

-- Trigger function for auto-creating profile on new auth.users entry & setting initial roles in auth.users

CREATE OR REPLACE FUNCTION public.handle\_new\_user()

RETURNS TRIGGER AS $$

DECLARE

default\_role\_code TEXT;

initial\_account\_status public.user\_account\_status\_enum := 'active';

new\_user\_email TEXT;

initial\_profile\_roles TEXT[];

BEGIN

new\_user\_email := NEW.email;

IF NEW.email\_confirmed\_at IS NULL AND NEW.phone\_confirmed\_at IS NULL THEN

initial\_account\_status := 'email\_unconfirmed';

END IF;

SELECT role\_code INTO default\_role\_code

FROM public.user\_roles\_master

WHERE default\_for\_new\_pilgrim\_users = TRUE AND is\_role\_active = TRUE AND deleted\_at IS NULL

LIMIT 1;

IF default\_role\_code IS NULL THEN

RAISE WARNING 'No default role found in user\_roles\_master. Falling back to ''pilgrim\_user''. Configure a default role.';

default\_role\_code := 'pilgrim\_user';

END IF;

initial\_profile\_roles := ARRAY[default\_role\_code]::text[];

-- Insert into public.profiles

INSERT INTO public.profiles (id, roles, preferred\_language\_code, account\_status, username, public\_display\_name, created\_at, updated\_at, last\_activity\_at) -- Added last\_activity\_at

VALUES (

NEW.id,

initial\_profile\_roles,

'en', -- Platform default language

initial\_account\_status,

COALESCE(NEW.raw\_user\_meta\_data->>'username', split\_part(new\_user\_email, '@', 1) || '\_' || substr(NEW.id::text, 1, 4)),

COALESCE(NEW.raw\_user\_meta\_data->>'full\_name', NEW.raw\_user\_meta\_data->>'name', split\_part(new\_user\_email, '@', 1)),

now(), -- created\_at

now(), -- updated\_at

now() -- last\_activity\_at set to now on creation

)

ON CONFLICT (id) DO NOTHING;

-- Update auth.users with the initial roles for JWT inclusion

UPDATE auth.users

SET raw\_app\_meta\_data = COALESCE(raw\_app\_meta\_data, '{}'::jsonb) || jsonb\_build\_object('roles', initial\_profile\_roles)

WHERE id = NEW.id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

COMMENT ON FUNCTION public.handle\_new\_user() IS 'Trigger function to automatically create a user profile in public.profiles when a new user signs up in auth.users. It sets initial default roles in both public.profiles.roles and auth.users.raw\_app\_meta\_data.roles, and initializes last\_activity\_at. Runs as SECURITY DEFINER.';

-- Apply the new user trigger to auth.users table

CREATE TRIGGER on\_auth\_user\_created

AFTER INSERT ON auth.users

FOR EACH ROW EXECUTE FUNCTION public.handle\_new\_user();

COMMENT ON TRIGGER on\_auth\_user\_created ON auth.users IS 'After a new user is created in auth.users, this trigger creates their corresponding profile and sets initial roles and activity timestamp.';

-- Conditional: Trigger function to clean up related translations for profiles

/\* (Definition as in V2.1/V2.2 spec) \*/

5. JSON Schema Mirror

{

"title": "profile",

"description": "Stores application-specific user profile data, extending Supabase Auth users. Manages identity, roles (synchronized to auth.users for JWT), preferences, contributor context, account status, activity timestamps, and admin update audit. Version 2.3.",

"type": "object",

"properties": {

"id": { "type": "string", "format": "uuid", "description": "User's ID from auth.users table (PK). Implicitly created\_by."},

"roles": { "type": "array", "items": { "type": "string" }, "description": "Array of role codes. Validated by trigger. Synchronized to auth.users.raw\_app\_meta\_data.roles. Cannot be empty.", "minItems": 1 },

"username": { "type": ["string", "null"], "description": "Optional unique username." },

"full\_name": { "type": ["string", "null"], "description": "User's full name." },

"public\_display\_name": { "type": ["string", "null"], "description": "Publicly shown name. Unique if provided." },

"public\_avatar\_media\_id": { "type": ["string", "null"], "format": "uuid", "description": "FK to media.id for public avatar." },

"public\_bio": { "type": ["string", "null"], "maxLength": 500, "description": "Short public biography (app-enforced max length)." },

"preferred\_language\_code": { "type": "string", "default": "en", "pattern": "^[a-z]{2}(-[A-Z]{2})?$", "description": "User's preferred language (IETF tag). FK to languages\_master." },

"preferred\_units\_of\_measure": { "$ref": "#/definitions/units\_preference\_enum", "default": "metric" },

"preferred\_timezone": { "type": ["string", "null"], "default": "Europe/Rome", "description": "IANA timezone string." },

"pilgrim\_experience\_level": { "$ref": "#/definitions/pilgrim\_experience\_enum" },

"pilgrimage\_interests\_tags": { "type": ["array", "null"], "items": { "type": "string" }, "description": "Array of interest tags." },

"contributor\_organization\_name": { "type": ["string", "null"], "description": "Organization name if contributing for one." },

"contributor\_organization\_role": { "type": ["string", "null"], "description": "User's role in that organization." },

"contact\_public\_email": { "type": ["string", "null"], "format": "email", "description": "Publicly visible email. Validated format." },

"website\_url\_profile": { "type": ["string", "null"], "format": "url", "description": "Personal/organizational website URL. Validated format." },

"notification\_preferences\_json": { "type": ["object", "null"], "description": "JSONB for notification preferences." },

"is\_profile\_publicly\_visible": { "type": "boolean", "default": false, "description": "Master switch for public visibility of profile." },

"contribution\_score": { "type": "integer", "default": 0, "minimum": 0, "description": "Score from positive contributions." },

"account\_status": { "$ref": "#/definitions/user\_account\_status\_enum", "default": "active" },

"terms\_accepted\_at": { "type": ["string", "null"], "format": "date-time", "description": "Timestamp of last terms acceptance." },

"last\_login\_at": { "type": ["string", "null"], "format": "date-time", "description": "Timestamp of user's last login (app-updated)." },

"last\_activity\_at": { "type": ["string", "null"], "format": "date-time", "description": "Timestamp of the last recorded user activity." },

"created\_at": { "type": "string", "format": "date-time", "description": "Timestamp of profile creation.", "readOnly": true },

"updated\_at": { "type": "string", "format": "date-time", "description": "Timestamp of last profile update.", "readOnly": true },

"updated\_by\_profile\_id": { "type": ["string", "null"], "format": "uuid", "description": "Profile ID of user (typically admin) who last updated this profile." }

},

"required": [ "id", "roles", "preferred\_language\_code", "preferred\_units\_of\_measure", "is\_profile\_publicly\_visible", "contribution\_score", "account\_status" ],

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

"unique\_constraints": [

{"columns": ["username"], "name": "uq\_profiles\_username", "allow\_null\_results": true},

{"columns": ["public\_display\_name"], "name": "uq\_profiles\_public\_display\_name", "allow\_null\_results": true}

],

"foreign\_keys": [

{"columns": ["id"], "references\_table": "auth.users", "references\_columns": ["id"], "on\_delete": "CASCADE"},

{"columns": ["public\_avatar\_media\_id"], "references\_table": "public.media", "references\_columns": ["id"], "on\_delete": "SET NULL"},

{"columns": ["preferred\_language\_code"], "references\_table": "public.languages\_master", "references\_columns": ["language\_code"], "on\_delete": "RESTRICT", "on\_update": "CASCADE"},

{"columns": ["updated\_by\_profile\_id"], "references\_table": "public.profiles", "references\_columns": ["id"], "on\_delete": "SET NULL"}

],

"definitions": {

"units\_preference\_enum": { "type": "string", "enum": ["metric", "imperial"] },

"pilgrim\_experience\_enum": { "type": ["string", "null"], "enum": [null, "novice\_first\_pilgrimage", "intermediate\_few\_pilgrimages", "experienced\_many\_pilgrimages", "long\_distance\_veteran"] },

"user\_account\_status\_enum": { "type": "string", "enum": ["active", "pending\_verification", "email\_unconfirmed", "suspended\_by\_admin", "deactivated\_by\_user"] }

}

}

6. Relationships & Integrity

Primary Key: `id` (uuid).

Foreign Keys: To `auth.users`, `media`, `languages\_master`, and self-referencing `updated\_by\_profile\_id`.

Array Foreign Key: `roles` (text[]) integrity with `user\_roles\_master` maintained by `check\_profile\_roles()` trigger.

Synchronization: `roles` are synchronized to `auth.users.raw\_app\_meta\_data.roles` via `handle\_new\_user()` and `sync\_profile\_roles\_to\_auth\_user()` triggers for JWT inclusion.

Mermaid ER Snippet:

erDiagram

auth\_users ||--o{ profiles : "has one"

profiles }o--|| media : "avatar (optional)"

profiles }o--|| languages\_master : "prefers lang"

profiles ..> user\_roles\_master : "roles from"

profiles }o--o| profiles : "updated\_by (admin)"

auth\_users { uuid id PK }

media { uuid id PK }

languages\_master { text language\_code PK }

user\_roles\_master { text role\_code PK }

profiles {

uuid id PK

text[] roles

uuid public\_avatar\_media\_id FK

text preferred\_language\_code FK

uuid updated\_by\_profile\_id FK

user\_account\_status\_enum account\_status

timestamptz last\_activity\_at

}

7. Multilingual Strategy

Translatable Fields: For V1, `public\_bio` is user-generated content stored in the language of submission. If it (or other fields like `full\_name` if used publicly) were to become centrally managed and translated, they would use `public.translations` with `table\_identifier: 'profiles'`, `column\_identifier: 'public\_bio'`, and `row\_foreign\_key: [profiles.id]`. An orphan-cleanup trigger on `profiles` would then be needed.

Driving Field: `preferred\_language\_code` signals the application for UI and content language.

8. Role-Based Workflow & RLS Notes

Key Fields for Roles/Workflows:

• `roles` (text[]): Core field determining user capabilities and access. Integrity enforced by `check\_profile\_roles()` trigger. Initial role(s) set by `handle\_new\_user()` trigger and synchronized to `auth.users.raw\_app\_meta\_data.roles`.

• `account\_status` (user\_account\_status\_enum): Manages user lifecycle.

• `last\_activity\_at`: Provides data for user engagement tracking.

Recommended Row-Level-Security (RLS) Policies:

• 🟢 Policy 1: Users can read their own profile.

```sql

CREATE POLICY "select\_own\_profile" ON public.profiles FOR SELECT USING (auth.uid() = id);

```

• 🟢 Policy 2: Users can update their own profile.

```sql

CREATE POLICY "update\_own\_profile" ON public.profiles FOR UPDATE USING (auth.uid() = id) WITH CHECK (auth.uid() = id);

```

• 🟠 Policy 3: Allow public read of specific profile aspects via a `public\_profiles\_view` (recommended) where `is\_profile\_publicly\_visible = true` and `account\_status = 'active'::public.user\_account\_status\_enum`.

• 🟢 Policy 4: Admins can manage all profiles (requires `public.has\_role\_on\_profile(UUID, TEXT)` helper).

```sql

CREATE POLICY "admin\_manage\_all\_profiles" ON public.profiles FOR ALL

USING (public.has\_role\_on\_profile(auth.uid(), 'admin\_platform') OR public.has\_role\_on\_profile(auth.uid(), 'admin\_super'))

WITH CHECK (public.has\_role\_on\_profile(auth.uid(), 'admin\_platform') OR public.has\_role\_on\_profile(auth.uid(), 'admin\_super'));

```

RLS may also leverage roles present in the JWT (from `auth.users.raw\_app\_meta\_data.roles`) for broader checks.

9. ENUM vs Lookup Discussion

🟢 `units\_preference\_enum`, `pilgrim\_experience\_enum`, `user\_account\_status\_enum` remain ENUMs for V1.

10. UI/UX Enablement

Columns powering UI: `preferred\_language\_code`, `public\_display\_name`, `public\_avatar\_media\_id`, `public\_bio`, `roles`, `last\_activity\_at`, etc.

Indexed Fields for UX Performance: GIN indexes on `roles`, `pilgrimage\_interests\_tags`. B-tree index on `account\_status`, `last\_activity\_at`.

11. Auditing & Lifecycle Management

Audit Columns: `created\_at`, `updated\_at`. `id` is implicit `created\_by`. `updated\_by\_profile\_id` tracks non-owner (admin) updates. `roles` changes are reflected in `auth.users.raw\_app\_meta\_data.roles`.

Lifecycle: `account\_status` enum. `last\_activity\_at` provides activity data. Hard delete cascades from `auth.users`.

12. Scalability & Future-Proofing

`roles` Array: Text array with GIN index is adequate.

Partitioning: Not foreseen as needed.

13. Seed Data

⚪ N/A - Profile records are created dynamically upon user registration.

14. Next-Action Checklist

• 🔴 Implement ENUM Types.

• 🔴 Create Table: Execute DDL for `public.profiles` (including `last\_activity\_at`).

• 🔴 Create Indexes (including for `last\_activity\_at`).

• 🔴 Implement `updated\_at` Trigger.

• 🔴 Implement Profile Creation & Role Sync Logic:

\* Update and apply `public.handle\_new\_user()` trigger on `auth.users` (to set `last\_activity\_at` and sync roles).

\* Create and apply `public.sync\_profile\_roles\_to\_auth\_user()` function and its trigger on `public.profiles`.

• 🔴 Implement `roles` Integrity Check: Create and apply `public.check\_profile\_roles()` trigger.

• 🟠 Test role synchronization mechanism thoroughly.

• 🟠 Define `updated\_by\_profile\_id` Population Logic for admin actions.

• 🟠 Define and implement update mechanism for `last\_activity\_at` (e.g., via API calls on significant user actions or specific DB triggers on related tables).

• 🟢 Consider Full-Text Search on profile text fields for V2+.

• 🟢 Conditional Translations Cleanup Trigger for `public\_bio` if it becomes system-translatable.