<https://gemini.google.com/u/1/app/9ed6b32c76b55d3a?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/8d62cfd96b565ac8>

<https://gemini.google.com/u/1/app/2d30425fc8a5984a>

\* \* \* \* \*

Okay, here is the single, fully complete "Updated Production-Ready Specification" covering segments, segment\_additional\_terrain\_types, and segment\_media, based on 2.4 segments.docx and our V2 review.

### **Updated Production-Ready Specification**

**Table Names:** public.segments, public.segment\_additional\_terrain\_types, public.segment\_media

**1. Purpose & Primary Use-Cases**

* public.segments: This table stores the most granular, fundamental, and reusable sections of a trail's path. Each segment is defined by a 3D geometry (path\_geom) connecting two distinct waypoints. Segments are the detailed building blocks for routes and user-facing "stages," central to mapping, navigation, elevation profiling, and detailed daily planning for pilgrims.
* public.segment\_additional\_terrain\_types: This junction table links a segment to multiple, non-dominant terrain\_types\_master records. It allows for a richer description of a segment's terrain beyond just its single dominant\_terrain\_type\_id.
* public.segment\_media: This junction table links a segment to multiple media records, creating a gallery for the segment. It supports ordering, semantic roles for media (e.g., 'gallery\_image', 'detail\_map'), and translatable captions and alt text for accessibility.

Key User-Story Touchpoints for segments and its related tables:

* Pilgrim (Anna) - Story A2 (Daily Stage Understanding): Provides core data (distance, time, elevation, path geometry, descriptions, media) for each piece of a daily stage.
* Pilgrim (Anna) - Story A4 (Identifying Key Services & POIs): segments.path\_geom allows spatial queries for nearby waypoints.
* Pilgrim (Anna) - Story A5 (Checking Trail Conditions): segments.operational\_status provides the most granular status.
* Regional Content Manager (Sofia) - Story C1 (Reporting Trail Warnings): Sofia updates segments.operational\_status or manages segment\_warnings.
* Platform Administrator (Admin Team) - Story D1 & D2: Admins manage segment definitions and their associated media/terrain details.

**2. Schema**

**Schema for** public.segments

| **Column Name** | **Data Type** | **Constraints** | **Description (Translatable fields marked)** |
| --- | --- | --- | --- |
| id | bigint | PRIMARY KEY, GENERATED ALWAYS AS IDENTITY | Unique identifier for each segment. |
| name | text | NOT NULL, CHECK (char\_length(name) &lt;= 255) | Descriptive name in English. (Translatable via public.translations) |
| slug | text | UNIQUE, CHECK (slug IS NULL OR (slug ~ '^[a-z0-9]+(?:-[a-z0-9]+)\*$' AND char\_length(slug) &lt;= 255)) | Optional URL-friendly identifier. |
| start\_waypoint\_id | bigint | NOT NULL, REFERENCES public.waypoints(id) ON DELETE RESTRICT | Waypoint marking the segment's beginning. |
| end\_waypoint\_id | bigint | NOT NULL, REFERENCES public.waypoints(id) ON DELETE RESTRICT | Waypoint marking the segment's end. |
| path\_geom | geometry(LineStringZ, 4326) | NOT NULL | Actual 3D geographical path (SRID 4326). |
| distance\_km | real | NOT NULL CHECK (distance\_km >= 0) | 🔴 Auto-calculated distance (km) from path\_geom. |
| estimated\_walking\_time\_minutes | integer | CHECK (estimated\_walking\_time\_minutes IS NULL OR estimated\_walking\_time\_minutes >= 0) | Estimated typical walking time (minutes). |
| elevation\_gain\_meters | integer | CHECK (elevation\_gain\_meters IS NULL OR elevation\_gain\_meters >= 0) | 🔴 Auto-calculated total ascent (m) from path\_geom. |
| elevation\_loss\_meters | integer | CHECK (elevation\_loss\_meters IS NULL OR elevation\_loss\_meters >= 0) | 🔴 Auto-calculated total descent (m) from path\_geom. |
| min\_elevation\_meters | integer |  | 🔴 Auto-calculated lowest elevation point (m) from path\_geom. |
| max\_elevation\_meters | integer |  | 🔴 Auto-calculated highest elevation point (m) from path\_geom. |
| average\_gradient\_percentage | real |  | 🔴 Optional: Auto-calculated average gradient. |
| elevation\_profile\_data | jsonb |  | 🔴 Stored array of [distance, elevation] points for charts. Auto-generated from path\_geom. |
| segment\_difficulty | public.trail\_difficulty\_enum |  | Difficulty specific to this segment. |
| dominant\_terrain\_type\_id | integer | REFERENCES public.terrain\_types\_master(id) ON DELETE SET NULL | FK to terrain\_types\_master for the predominant terrain type. |
| sun\_exposure\_level | public.segment\_sun\_exposure\_enum |  | General sun exposure on this segment. |
| recommended\_travel\_direction | public.segment\_travel\_direction\_enum |  | Recommended or mandatory direction of travel. |
| operational\_status | public.trail\_operational\_status\_enum | NOT NULL DEFAULT 'information\_unavailable' | Operational status for this specific segment. |
| content\_visibility\_status | public.content\_visibility\_status\_enum | NOT NULL DEFAULT 'draft' | Editorial status for public view. |
| is\_detour\_for\_segment\_id | bigint | REFERENCES public.segments(id) ON DELETE SET NULL | If a detour, links to the ID of the original segment it bypasses. |
| gpx\_media\_id | bigint | REFERENCES public.media(id) ON DELETE SET NULL | Optional FK to public.media for a GPX file (media\_asset\_type='gpx\_file'). Replaces gpx\_track\_data\_url. |
| short\_description | text |  | Brief overview in English. (Translatable via public.translations) |
| detailed\_description\_notes | text |  | In-depth narrative notes in English. (Translatable via public.translations) |
| waymarking\_on\_segment\_notes | text |  | Specific waymarking details in English. (Translatable via public.translations) |
| segment\_suitability\_notes | text |  | Suitability notes in English. (Translatable via public.translations) |
| water\_sources\_general\_notes | text |  | General commentary on water availability in English. (Translatable via public.translations) |
| resupply\_options\_general\_notes | text |  | General commentary on food/resupply in English. (Translatable via public.translations) |
| segment\_cultural\_historical\_notes | text |  | Cultural/historical notes for this segment's path in English. (Translatable via public.translations) |
| emergency\_access\_notes | text |  | Emergency access notes in English. (Translatable via public.translations) |
| segment\_weather\_advice | text |  | Weather-related advice in English. (Translatable via public.translations) |
| primary\_data\_source\_segment | text |  | Source of this specific segment's data. |
| created\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp of record creation. |
| created\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who created this segment. |
| updated\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp of last update (auto-updated by trigger). |
| updated\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who last updated this segment. |
| deleted\_at | timestamptz |  | Timestamp for soft deletion. |

**Schema for** public.segment\_additional\_terrain\_types

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| segment\_id | bigint | PRIMARY KEY (Component), NOT NULL, REFERENCES public.segments(id) ON DELETE CASCADE | The ID of the segment. |
| terrain\_type\_id | integer | PRIMARY KEY (Component), NOT NULL, REFERENCES public.terrain\_types\_master(id) ON DELETE RESTRICT | The ID of the additional terrain type (from terrain\_types\_master). |
| created\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp indicating when this specific segment-terrain link was created. |
| created\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who created this link. |
| updated\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp indicating when this link record was last updated (auto-updated by trigger). |
| updated\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who last updated this link. |

**Schema for** public.segment\_media

| **Column Name** | **Data Type** | **Constraints** | **Description (Translatable fields marked)** |
| --- | --- | --- | --- |
| id | bigint | PRIMARY KEY, GENERATED ALWAYS AS IDENTITY | Unique identifier for the segment-media link. |
| segment\_id | bigint | NOT NULL, REFERENCES public.segments(id) ON DELETE CASCADE | The ID of the segment this media belongs to. |
| media\_id | bigint | NOT NULL, REFERENCES public.media(id) ON DELETE CASCADE | The ID of the media item. |
| media\_role\_code | text | REFERENCES public.media\_roles\_master(code) ON DELETE SET NULL | Semantic role of the media (e.g., 'gallery\_image'). |
| display\_order | integer | NOT NULL DEFAULT 0 | Order for displaying media within a segment's gallery, per role. |
| caption | text |  | Caption for the media in English. (Translatable via public.translations) |
| alt\_text | text |  | Alternative text for the media, for accessibility, in English. (Translatable via public.translations) |
| created\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp of record creation. |
| created\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who created this link. |
| updated\_at | timestamptz | NOT NULL DEFAULT now() | Timestamp of last update (auto-updated by trigger). |
| updated\_by\_profile\_id | uuid | REFERENCES public.profiles(id) ON DELETE SET NULL | Profile ID of the user who last updated this link. |
|  |  | UNIQUE (segment\_id, media\_id, media\_role\_code) | Ensures a media item is not linked multiple times to the same segment with the same role. (Updated for role) |
|  |  | UNIQUE (segment\_id, display\_order, media\_role\_code) | Ensures display order is unique within a segment's gallery for a given role. (Updated for role) |

**3. PostgreSQL DDL (**segments**,** segment\_additional\_terrain\_types**,** segment\_media**)**

SQL

-- ENUM Types

-- Assuming 'public.trail\_difficulty\_enum', 'public.trail\_operational\_status\_enum',

-- 'public.content\_visibility\_status\_enum' are already created.

DO $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM pg\_type WHERE typname = 'segment\_sun\_exposure\_enum') THEN

CREATE TYPE public.segment\_sun\_exposure\_enum AS ENUM (

'mostly\_shaded',

'partially\_shaded',

'mostly\_exposed',

'variable'

); --

END IF;

IF NOT EXISTS (SELECT 1 FROM pg\_type WHERE typname = 'segment\_travel\_direction\_enum') THEN

CREATE TYPE public.segment\_travel\_direction\_enum AS ENUM (

'bidirectional',

'northbound\_only',

'southbound\_only',

'eastbound\_only',

'westbound\_only',

'uphill\_only',

'downhill\_only',

'clockwise\_only',

'counter\_clockwise\_only',

'as\_signposted'

); --

END IF;

END$$;

-- Assuming 'public.waypoints', 'public.terrain\_types\_master', 'public.media', 'public.profiles', 'public.media\_roles\_master' tables exist.

-- Function to calculate geometric properties from path\_geom

CREATE OR REPLACE FUNCTION public.calculate\_segment\_geom\_properties(p\_path\_geom geometry)

RETURNS TABLE(

out\_distance\_km real,

out\_elevation\_gain\_meters integer,

out\_elevation\_loss\_meters integer,

out\_min\_elevation\_meters integer,

out\_max\_elevation\_meters integer,

out\_average\_gradient\_percentage real,

out\_elevation\_profile\_data jsonb

) AS $$

DECLARE

num\_points integer;

current\_elevation double precision;

prev\_elevation double precision;

total\_dist\_3d double precision;

total\_gain double precision := 0;

total\_loss double precision := 0;

min\_elev double precision := NULL;

max\_elev double precision := NULL;

segment\_length\_2d double precision;

profile\_points jsonb[] := '{}';

point\_interval real := 50; -- meters, interval for profile points

current\_dist\_along real := 0;

temp\_point geometry;

interpolated\_z double precision;

BEGIN

IF p\_path\_geom IS NULL OR ST\_IsEmpty(p\_path\_geom) OR ST\_SRID(p\_path\_geom) != 4326 THEN

RETURN QUERY SELECT 0::real, 0, 0, NULL::integer, NULL::integer, NULL::real, '[]'::jsonb;

RETURN;

END IF;

total\_dist\_3d := ST\_3DLENGTH(p\_path\_geom);

num\_points := ST\_NPoints(p\_path\_geom);

IF num\_points < 2 THEN

prev\_elevation := ST\_Z(ST\_PointN(p\_path\_geom, 1));

RETURN QUERY SELECT 0::real, 0, 0, prev\_elevation::integer, prev\_elevation::integer, 0::real, jsonb\_build\_array(jsonb\_build\_array(0, round(prev\_elevation::numeric, 2)));

RETURN;

END IF;

prev\_elevation := ST\_Z(ST\_PointN(p\_path\_geom, 1));

min\_elev := prev\_elevation;

max\_elev := prev\_elevation;

profile\_points := array\_append(profile\_points, jsonb\_build\_array(0, round(prev\_elevation::numeric, 2)));

FOR i IN 2..num\_points LOOP

current\_elevation := ST\_Z(ST\_PointN(p\_path\_geom, i));

IF current\_elevation > prev\_elevation THEN

total\_gain := total\_gain + (current\_elevation - prev\_elevation);

ELSIF current\_elevation < prev\_elevation THEN

total\_loss := total\_loss + (prev\_elevation - current\_elevation);

END IF;

min\_elev := least(min\_elev, current\_elevation);

max\_elev := greatest(max\_elev, current\_elevation);

prev\_elevation := current\_elevation;

END LOOP;

current\_dist\_along := point\_interval;

WHILE current\_dist\_along < total\_dist\_3d LOOP

temp\_point := ST\_LineInterpolatePoint(p\_path\_geom, current\_dist\_along / total\_dist\_3d);

interpolated\_z := ST\_Z(temp\_point);

profile\_points := array\_append(profile\_points, jsonb\_build\_array(round(current\_dist\_along::numeric, 2), round(interpolated\_z::numeric, 2)));

current\_dist\_along := current\_dist\_along + point\_interval;

END LOOP;

profile\_points := array\_append(profile\_points, jsonb\_build\_array(round(total\_dist\_3d::numeric, 2), round(ST\_Z(ST\_PointN(p\_path\_geom, num\_points))::numeric, 2)));

SELECT array\_agg(DISTINCT el) INTO profile\_points FROM unnest(profile\_points) el;

segment\_length\_2d := ST\_Length(ST\_Force2D(p\_path\_geom));

out\_average\_gradient\_percentage := CASE

WHEN segment\_length\_2d > 0 THEN round(((max\_elev - min\_elev) / segment\_length\_2d \* 100)::numeric, 2)

ELSE 0

END;

RETURN QUERY SELECT

round((total\_dist\_3d / 1000)::numeric, 3)::real,

round(total\_gain::numeric)::integer,

round(total\_loss::numeric)::integer,

round(min\_elev::numeric)::integer,

round(max\_elev::numeric)::integer,

out\_average\_gradient\_percentage::real,

array\_to\_json(profile\_points)::jsonb;

END;

$$ LANGUAGE plpgsql IMMUTABLE STRICT; --

-- Trigger function to update segment properties from path\_geom

CREATE OR REPLACE FUNCTION public.update\_segment\_geom\_derived\_fields()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' OR (TG\_OP = 'UPDATE' AND NEW.path\_geom IS DISTINCT FROM OLD.path\_geom) THEN

SELECT

props.out\_distance\_km,

props.out\_elevation\_gain\_meters,

props.out\_elevation\_loss\_meters,

props.out\_min\_elevation\_meters,

props.out\_max\_elevation\_meters,

props.out\_average\_gradient\_percentage,

props.out\_elevation\_profile\_data

INTO

NEW.distance\_km,

NEW.elevation\_gain\_meters,

NEW.elevation\_loss\_meters,

NEW.min\_elevation\_meters,

NEW.max\_elevation\_meters,

NEW.average\_gradient\_percentage,

NEW.elevation\_profile\_data

FROM public.calculate\_segment\_geom\_properties(NEW.path\_geom) props; --

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql; --

CREATE TABLE public.segments (

id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

name TEXT NOT NULL CHECK (char\_length(name) <= 255),

slug TEXT UNIQUE CHECK (slug IS NULL OR (slug ~ '^[a-z0-9]+(?:-[a-z0-9]+)\*$' AND char\_length(slug) <= 255)),

start\_waypoint\_id BIGINT NOT NULL REFERENCES public.waypoints(id) ON DELETE RESTRICT,

end\_waypoint\_id BIGINT NOT NULL REFERENCES public.waypoints(id) ON DELETE RESTRICT,

path\_geom GEOMETRY(LineStringZ, 4326) NOT NULL,

distance\_km REAL NOT NULL CHECK (distance\_km >= 0),

estimated\_walking\_time\_minutes INTEGER CHECK (estimated\_walking\_time\_minutes IS NULL OR estimated\_walking\_time\_minutes >= 0),

elevation\_gain\_meters INTEGER CHECK (elevation\_gain\_meters IS NULL OR elevation\_gain\_meters >= 0),

elevation\_loss\_meters INTEGER CHECK (elevation\_loss\_meters IS NULL OR elevation\_loss\_meters >= 0),

min\_elevation\_meters INTEGER,

max\_elevation\_meters INTEGER,

average\_gradient\_percentage REAL,

elevation\_profile\_data JSONB,

segment\_difficulty public.trail\_difficulty\_enum,

dominant\_terrain\_type\_id INTEGER REFERENCES public.terrain\_types\_master(id) ON DELETE SET NULL,

sun\_exposure\_level public.segment\_sun\_exposure\_enum,

recommended\_travel\_direction public.segment\_travel\_direction\_enum,

operational\_status public.trail\_operational\_status\_enum NOT NULL DEFAULT 'information\_unavailable',

content\_visibility\_status public.content\_visibility\_status\_enum NOT NULL DEFAULT 'draft',

is\_detour\_for\_segment\_id BIGINT REFERENCES public.segments(id) ON DELETE SET NULL,

gpx\_media\_id BIGINT REFERENCES public.media(id) ON DELETE SET NULL, -- V2 Update

short\_description TEXT,

detailed\_description\_notes TEXT,

waymarking\_on\_segment\_notes TEXT,

segment\_suitability\_notes TEXT,

water\_sources\_general\_notes TEXT,

resupply\_options\_general\_notes TEXT,

segment\_cultural\_historical\_notes TEXT,

emergency\_access\_notes TEXT,

segment\_weather\_advice TEXT,

primary\_data\_source\_segment TEXT,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

deleted\_at TIMESTAMPTZ,

CONSTRAINT chk\_start\_end\_different CHECK (start\_waypoint\_id <> end\_waypoint\_id) --

); --

-- Trigger for auto-calculating geometric properties

CREATE TRIGGER trigger\_segments\_geom\_derived\_fields

BEFORE INSERT OR UPDATE OF path\_geom ON public.segments

FOR EACH ROW

EXECUTE FUNCTION public.update\_segment\_geom\_derived\_fields(); --

-- Trigger for standard audit fields

CREATE OR REPLACE FUNCTION public.set\_segment\_modification\_meta()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'UPDATE' AND (NEW.path\_geom IS NOT DISTINCT FROM OLD.path\_geom) THEN

NEW.updated\_at = NOW();

END IF;

IF (TG\_OP = 'INSERT') THEN

NEW.created\_by\_profile\_id = auth.uid();

NEW.updated\_by\_profile\_id = auth.uid();

IF NEW.updated\_at IS NULL THEN NEW.updated\_at = NOW(); END IF;

ELSIF (TG\_OP = 'UPDATE') THEN

NEW.updated\_by\_profile\_id = auth.uid();

NEW.created\_at = OLD.created\_at;

NEW.created\_by\_profile\_id = OLD.created\_by\_profile\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER; --

CREATE TRIGGER trigger\_segments\_modification\_meta

BEFORE INSERT OR UPDATE ON public.segments

FOR EACH ROW

EXECUTE FUNCTION public.set\_segment\_modification\_meta(); --

-- Indexes for segments

CREATE INDEX IF NOT EXISTS idx\_segments\_slug ON public.segments(slug); --

CREATE INDEX IF NOT EXISTS idx\_segments\_start\_waypoint\_id ON public.segments(start\_waypoint\_id); --

CREATE INDEX IF NOT EXISTS idx\_segments\_end\_waypoint\_id ON public.segments(end\_waypoint\_id); --

CREATE INDEX IF NOT EXISTS idx\_segments\_path\_geom ON public.segments USING GIST (path\_geom); --

CREATE INDEX IF NOT EXISTS idx\_segments\_operational\_status ON public.segments(operational\_status); --

CREATE INDEX IF NOT EXISTS idx\_segments\_content\_visibility\_status ON public.segments(content\_visibility\_status); --

CREATE INDEX IF NOT EXISTS idx\_segments\_is\_detour\_for\_segment\_id ON public.segments(is\_detour\_for\_segment\_id); --

CREATE INDEX IF NOT EXISTS idx\_segments\_dominant\_terrain\_type\_id ON public.segments(dominant\_terrain\_type\_id); --

CREATE INDEX IF NOT EXISTS idx\_segments\_gpx\_media\_id ON public.segments(gpx\_media\_id); -- V2 Update

CREATE INDEX IF NOT EXISTS idx\_segments\_deleted\_at ON public.segments(deleted\_at) WHERE deleted\_at IS NULL; --

-- Comments for segments

COMMENT ON TABLE public.segments IS 'Stores the most granular, fundamental, and reusable sections of a trail path, defined by 3D geometry connecting two waypoints. Building blocks for routes and stages. Version: V2.'; --

COMMENT ON COLUMN public.segments.name IS 'Descriptive name of the segment in English (e.g., "La Verna Sanctuary to Chiusi Town"). Max 255 chars. (Translatable via public.translations). Version: V2.'; --

COMMENT ON COLUMN public.segments.path\_geom IS 'Actual 3D geographical path (LineStringZ with Longitude, Latitude, Altitude coordinates). Stored with SRID 4326. Requires PostGIS extension enabled. Version: V2.'; --

COMMENT ON COLUMN public.segments.distance\_km IS '🔴 Auto-calculated 3D distance of the segment in kilometers, derived from path\_geom. Version: V2.'; --

COMMENT ON COLUMN public.segments.gpx\_media\_id IS 'Foreign key to public.media table for the segment''s GPX track file (media\_asset\_type=''gpx\_file''). Replaces gpx\_track\_data\_url. Version: V2.';

COMMENT ON COLUMN public.segments.deleted\_at IS 'Timestamp for soft deletion. If NOT NULL, the segment is considered deleted/inactive. Version: V2.'; --

-- (Ensure all other comments from the previous full spec for segments are included here)

-- DDL for public.segment\_additional\_terrain\_types

CREATE TABLE public.segment\_additional\_terrain\_types (

segment\_id BIGINT NOT NULL REFERENCES public.segments(id) ON DELETE CASCADE,

terrain\_type\_id INTEGER NOT NULL REFERENCES public.terrain\_types\_master(id) ON DELETE RESTRICT, --

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

PRIMARY KEY (segment\_id, terrain\_type\_id)

); --

CREATE INDEX IF NOT EXISTS idx\_satt\_segment\_id ON public.segment\_additional\_terrain\_types(segment\_id);

CREATE INDEX IF NOT EXISTS idx\_satt\_terrain\_type\_id ON public.segment\_additional\_terrain\_types(terrain\_type\_id);

CREATE OR REPLACE FUNCTION public.set\_segment\_additional\_terrain\_type\_modification\_meta()

RETURNS TRIGGER AS $$

BEGIN

IF (TG\_OP = 'INSERT') THEN

NEW.created\_by\_profile\_id = auth.uid(); --

NEW.updated\_at = NOW(); --

NEW.updated\_by\_profile\_id = auth.uid(); --

ELSIF (TG\_OP = 'UPDATE') THEN

NEW.updated\_at = NOW(); --

NEW.updated\_by\_profile\_id = auth.uid(); --

NEW.created\_at = OLD.created\_at; --

NEW.created\_by\_profile\_id = OLD.created\_by\_profile\_id; --

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER; --

CREATE TRIGGER trigger\_segment\_additional\_terrain\_types\_modification\_meta

BEFORE INSERT OR UPDATE ON public.segment\_additional\_terrain\_types

FOR EACH ROW

EXECUTE FUNCTION public.set\_segment\_additional\_terrain\_type\_modification\_meta(); --

COMMENT ON TABLE public.segment\_additional\_terrain\_types IS 'Links segments to additional (non-dominant) terrain types. Version: V2.'; --

-- (Ensure all column comments for segment\_additional\_terrain\_types are included)

-- DDL for public.segment\_media

CREATE TABLE public.segment\_media (

id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

segment\_id BIGINT NOT NULL REFERENCES public.segments(id) ON DELETE CASCADE,

media\_id BIGINT NOT NULL REFERENCES public.media(id) ON DELETE CASCADE,

media\_role\_code TEXT REFERENCES public.media\_roles\_master(code) ON DELETE SET NULL,

display\_order INTEGER NOT NULL DEFAULT 0,

caption TEXT, -- (Translatable)

alt\_text TEXT, -- (Translatable)

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

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

UNIQUE (segment\_id, media\_id, media\_role\_code),

UNIQUE (segment\_id, display\_order, media\_role\_code)

); --

CREATE INDEX IF NOT EXISTS idx\_segment\_media\_segment\_id ON public.segment\_media(segment\_id);

CREATE INDEX IF NOT EXISTS idx\_segment\_media\_media\_id ON public.segment\_media(media\_id);

CREATE INDEX IF NOT EXISTS idx\_segment\_media\_media\_role\_code ON public.segment\_media(media\_role\_code);

CREATE OR REPLACE FUNCTION public.set\_segment\_media\_modification\_meta()

RETURNS TRIGGER AS $$

BEGIN

IF (TG\_OP = 'INSERT') THEN

NEW.created\_by\_profile\_id = auth.uid(); --

NEW.updated\_at = NOW(); --

NEW.updated\_by\_profile\_id = auth.uid(); --

ELSIF (TG\_OP = 'UPDATE') THEN

NEW.updated\_at = NOW(); --

NEW.updated\_by\_profile\_id = auth.uid(); --

NEW.created\_at = OLD.created\_at; --

NEW.created\_by\_profile\_id = OLD.created\_by\_profile\_id; --

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER; --

CREATE TRIGGER trigger\_segment\_media\_modification\_meta

BEFORE INSERT OR UPDATE ON public.segment\_media

FOR EACH ROW

EXECUTE FUNCTION public.set\_segment\_media\_modification\_meta(); --

COMMENT ON TABLE public.segment\_media IS 'Links segments to media items (gallery), defining order, role, and translatable captions/alt\_text. Version: V2.'; --

-- (Ensure all column comments for segment\_media are included)

**4. JSON Schema Mirror**

**JSON Schema for** public.segments

JSON

{

"title": "segment",

"description": "A granular, walkable section of a route, connecting two waypoints, with detailed path geometry and characteristics. It's a fundamental building block for constructing routes and displaying daily stages. Version: V2.",

"type": "object",

"properties": {

"id": { "type": "integer", "format": "int64", "description": "Unique identifier for the segment. Read-only.", "readOnly": true },

"name": { "type": "string", "maxLength": 255, "description": "Descriptive name of the segment in English. (Translatable via public.translations)" },

"slug": { "type": ["string", "null"], "maxLength": 255, "pattern": "^[a-z0-9]+(?:-[a-z0-9]+)\*$", "description": "Optional URL-friendly identifier for the segment." },

"start\_waypoint\_id": { "type": "integer", "format": "int64", "description": "Foreign key referencing the ID of the waypoint (public.waypoints.id) that marks the segment's beginning." },

"end\_waypoint\_id": { "type": "integer", "format": "int64", "description": "Foreign key referencing the ID of the waypoint (public.waypoints.id) that marks the segment's end." },

"path\_geom": { "type": "object", "description": "GeoJSON LineStringZ object representing the 3D geographical path (SRID 4326). Requires PostGIS. Source of truth for geometric calculations." },

"distance\_km": { "type": "number", "format": "float", "minimum": 0, "description": "🔴 Auto-calculated 3D distance of the segment in kilometers, derived from path\_geom. Read-only.", "readOnly": true },

"estimated\_walking\_time\_minutes": { "type": ["integer", "null"], "minimum": 0, "description": "Estimated typical walking time in minutes for this segment. May be manually entered or derived." },

"elevation\_gain\_meters": { "type": ["integer", "null"], "minimum": 0, "description": "🔴 Auto-calculated total ascent (cumulative positive elevation change) in meters, derived from path\_geom. Read-only.", "readOnly": true },

"elevation\_loss\_meters": { "type": ["integer", "null"], "minimum": 0, "description": "🔴 Auto-calculated total descent (cumulative negative elevation change) in meters, derived from path\_geom. Read-only.", "readOnly": true },

"min\_elevation\_meters": { "type": ["integer", "null"], "description": "🔴 Auto-calculated minimum elevation point in meters along the segment, derived from path\_geom. Read-only.", "readOnly": true },

"max\_elevation\_meters": { "type": ["integer", "null"], "description": "🔴 Auto-calculated maximum elevation point in meters along the segment, derived from path\_geom. Read-only.", "readOnly": true },

"average\_gradient\_percentage": { "type": ["number", "null"], "format": "float", "description": "🔴 Optional: Auto-calculated average gradient of the segment as a percentage. Derived from path\_geom. Read-only.", "readOnly": true },

"elevation\_profile\_data": { "type": ["object", "null"], "description": "🔴 JSONB array of [distance\_along\_segment, elevation] points. Auto-generated from path\_geom for rendering elevation profile charts. Read-only.", "readOnly": true },

"segment\_difficulty": { "type": ["string", "null"], "enum": ["easy", "moderate", "challenging", "strenuous", "variable", null], "description": "Subjective difficulty rating specific to this segment, reusing public.trail\_difficulty\_enum." },

"dominant\_terrain\_type\_id": { "type": ["integer", "null"], "format": "int32", "description": "Foreign key to public.terrain\_types\_master for the single most predominant terrain type on this segment." },

"sun\_exposure\_level": { "type": ["string", "null"], "enum": ["mostly\_shaded", "partially\_shaded", "mostly\_exposed", "variable", null], "description": "General sun exposure level expected on this segment, using public.segment\_sun\_exposure\_enum." },

"recommended\_travel\_direction": { "type": ["string", "null"], "enum": ["bidirectional", "northbound\_only", "southbound\_only", "eastbound\_only", "westbound\_only", "uphill\_only", "downhill\_only", "clockwise\_only", "counter\_clockwise\_only", "as\_signposted", null], "description": "Recommended or mandatory direction of travel for this segment, using public.segment\_travel\_direction\_enum." },

"operational\_status": { "type": "string", "enum": ["fully\_operational", "partially\_closed\_detours\_in\_place", "seasonal\_access\_only", "under\_development", "closed", "information\_unavailable"], "default": "information\_unavailable", "description": "Operational status for this specific segment, reusing public.trail\_operational\_status\_enum." },

"content\_visibility\_status": { "type": "string", "enum": ["draft", "pending\_review", "published", "archived"], "default": "draft", "description": "Editorial status indicating if the segment content is ready for public view, reusing public.content\_visibility\_status\_enum." },

"is\_detour\_for\_segment\_id": { "type": ["integer", "null"], "format": "int64", "description": "If this segment is a detour, this field links to the ID of the original segment (public.segments.id) it bypasses." },

"gpx\_media\_id": { "type": ["integer", "null"], "format": "int64", "description": "Foreign key to public.media for the segment's GPX track file (media\_asset\_type='gpx\_file')." },

"short\_description": { "type": ["string", "null"], "description": "Brief overview of the segment in English. (Translatable via public.translations)" },

"detailed\_description\_notes": { "type": ["string", "null"], "description": "In-depth narrative notes about the segment in English. (Translatable via public.translations)" },

"waymarking\_on\_segment\_notes": { "type": ["string", "null"], "description": "Specific waymarking details for navigating this segment in English. (Translatable via public.translations)" },

"segment\_suitability\_notes": { "type": ["string", "null"], "description": "Notes on suitability for different types of users or conditions. (Translatable via public.translations)" },

"water\_sources\_general\_notes": { "type": ["string", "null"], "description": "General commentary on water availability along this segment. (Translatable via public.translations)" },

"resupply\_options\_general\_notes": { "type": ["string", "null"], "description": "General commentary on food resupply options relevant to this segment. (Translatable via public.translations)" },

"segment\_cultural\_historical\_notes": { "type": ["string", "null"], "description": "Cultural or historical notes specific to this segment's path. (Translatable via public.translations)" },

"emergency\_access\_notes": { "type": ["string", "null"], "description": "Notes on emergency access points or procedures. (Translatable via public.translations)" },

"segment\_weather\_advice": { "type": ["string", "null"], "description": "Weather-related advice specific to this segment. (Translatable via public.translations)" },

"primary\_data\_source\_segment": { "type": ["string", "null"], "description": "Source of this specific segment's data (e.g., 'Official Park GPX Q1/2024')." },

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

"created\_by\_profile\_id": { "type": ["string", "null"], "format": "uuid", "description": "Profile ID (public.profiles.id) of the user who created this segment record. Read-only.", "readOnly": true },

"updated\_at": { "type": "string", "format": "date-time", "description": "Timestamp of the last update to this segment record. Read-only.", "readOnly": true },

"updated\_by\_profile\_id": { "type": ["string", "null"], "format": "uuid", "description": "Profile ID (public.profiles.id) of the user who last updated this segment record. Read-only.", "readOnly": true },

"deleted\_at": { "type": ["string", "null"], "format": "date-time", "description": "Timestamp for soft deletion. If present, the segment is considered inactive. Read-only.", "readOnly": true }

},

"required": [

"name",

"start\_waypoint\_id",

"end\_waypoint\_id",

"path\_geom",

"distance\_km",

"operational\_status",

"content\_visibility\_status",

"created\_at",

"updated\_at"

],

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

"unique\_constraints": [

{"columns": ["slug"], "name": "segments\_slug\_key"}

]

}

**JSON Schema for** public.segment\_additional\_terrain\_types

JSON

{

"title": "segment\_additional\_terrain\_type\_link",

"description": "Links a segment to its additional (non-dominant) terrain types. Many-to-many relationship. Version: V2.",

"type": "object",

"properties": {

"segment\_id": {

"type": "integer",

"format": "int64",

"description": "Foreign key referencing the ID of the segment (public.segments.id)."

},

"terrain\_type\_id": {

"type": "integer",

"format": "int32",

"description": "Foreign key referencing the ID of the terrain type from public.terrain\_types\_master."

},

"created\_at": {

"type": "string",

"format": "date-time",

"description": "Timestamp of when this link record was created. Read-only.",

"readOnly": true

},

"created\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID (public.profiles.id) of the user who created this link. Read-only.",

"readOnly": true

},

"updated\_at": {

"type": "string",

"format": "date-time",

"description": "Timestamp of when this link record was last updated. Read-only.",

"readOnly": true

},

"updated\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID (public.profiles.id) of the user who last updated this link. Read-only.",

"readOnly": true

}

},

"required": [

"segment\_id",

"terrain\_type\_id",

"created\_at",

"updated\_at"

],

"primary\_key": ["segment\_id", "terrain\_type\_id"]

}

**JSON Schema for** public.segment\_media

JSON

{

"title": "segment\_media\_link",

"description": "Links a segment to media items (e.g., for a photo gallery), defining display order, role, and translatable captions/alt\_text. Version: V2.",

"type": "object",

"properties": {

"id": {

"type": "integer",

"format": "int64",

"description": "Unique identifier for the segment-media link record. Read-only.",

"readOnly": true

},

"segment\_id": {

"type": "integer",

"format": "int64",

"description": "Foreign key referencing the ID of the segment (public.segments.id)."

},

"media\_id": {

"type": "integer",

"format": "int64",

"description": "Foreign key referencing the ID of the media item (public.media.id)."

},

"media\_role\_code": {

"type": ["string", "null"],

"description": "Semantic role of the media in relation to the segment (e.g., 'gallery\_image', 'detail\_map'). FK to public.media\_roles\_master.code."

},

"display\_order": {

"type": "integer",

"default": 0,

"description": "Order for displaying media within a segment's gallery, per role."

},

"caption": {

"type": ["string", "null"],

"description": "Caption for the media in English. (Translatable via public.translations)"

},

"alt\_text": {

"type": ["string", "null"],

"description": "Alternative text for the media, for accessibility, in English. (Translatable via public.translations)"

},

"created\_at": {

"type": "string",

"format": "date-time",

"description": "Timestamp of record creation. Read-only.",

"readOnly": true

},

"created\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID (public.profiles.id) of the user who created this link. Read-only.",

"readOnly": true

},

"updated\_at": {

"type": "string",

"format": "date-time",

"description": "Timestamp of last update. Read-only.",

"readOnly": true

},

"updated\_by\_profile\_id": {

"type": ["string", "null"],

"format": "uuid",

"description": "Profile ID (public.profiles.id) of the user who last updated this link. Read-only.",

"readOnly": true

}

},

"required": [

"segment\_id",

"media\_id",

"display\_order",

"created\_at",

"updated\_at"

],

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

"unique\_constraints": [

{"columns": ["segment\_id", "media\_id", "media\_role\_code"], "name": "segment\_media\_segment\_id\_media\_id\_media\_role\_code\_key"},

{"columns": ["segment\_id", "display\_order", "media\_role\_code"], "name": "segment\_media\_segment\_id\_display\_order\_media\_role\_code\_key"}

]

}

**5. Relationships & Integrity**

* segments **Table FKs:**
  + start\_waypoint\_id references public.waypoints(id): ON DELETE RESTRICT.
  + end\_waypoint\_id references public.waypoints(id): ON DELETE RESTRICT.
  + dominant\_terrain\_type\_id references public.terrain\_types\_master(id): ON DELETE SET NULL.
  + is\_detour\_for\_segment\_id references public.segments(id): ON DELETE SET NULL.
  + gpx\_media\_id references public.media(id): ON DELETE SET NULL. (V2 Update)
  + Audit FKs to public.profiles(id): ON DELETE SET NULL.
* segment\_additional\_terrain\_types **Junction Table:**
  + segment\_id references public.segments(id): ON DELETE CASCADE.
  + terrain\_type\_id references public.terrain\_types\_master(id): ON DELETE RESTRICT.
  + Audit FKs to public.profiles(id): ON DELETE SET NULL.
* segment\_media **Junction Table:**
  + segment\_id references public.segments(id): ON DELETE CASCADE.
  + media\_id references public.media(id): ON DELETE CASCADE.
  + media\_role\_code references public.media\_roles\_master(code): ON DELETE SET NULL. (V2 Update)
  + Audit FKs to public.profiles(id): ON DELETE SET NULL.
* **Constraint:** chk\_start\_end\_different on segments ensures start\_waypoint\_id <> end\_waypoint\_id.
* **Mermaid ER Diagram Snippet:**
* Code snippet

erDiagram

segments {

bigint id PK

text name

bigint start\_waypoint\_id FK

bigint end\_waypoint\_id FK

geometry\_LineStringZ path\_geom

real distance\_km

integer elevation\_gain\_meters

integer elevation\_loss\_meters

integer dominant\_terrain\_type\_id FK

bigint is\_detour\_for\_segment\_id FK

bigint gpx\_media\_id FK

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

timestamptz deleted\_at

}

waypoints { bigint id PK }

terrain\_types\_master { integer id PK }

media { bigint id PK }

media\_roles\_master { text code PK }

profiles { uuid id PK }

segment\_additional\_terrain\_types {

bigint segment\_id PK FK

integer terrain\_type\_id PK FK

}

segment\_media {

bigint id PK

bigint segment\_id FK

bigint media\_id FK

text media\_role\_code FK

}

routes { bigint id PK } -- For RLS context

route\_segments { -- For RLS context

bigint route\_id FK

bigint segment\_id FK

}

trails { bigint id PK } -- For RLS context

segments }o--|| waypoints : "starts\_at (RESTRICT)"

segments }o--|| waypoints : "ends\_at (RESTRICT)"

segments }o--o| segments : "is\_detour\_for (SET NULL)"

segments ||--o| terrain\_types\_master : "has\_dominant\_terrain (SET NULL)"

segments ||--o| media : "has\_gpx\_track (SET NULL)"

segments ||--o| profiles : "created\_by (SET NULL)"

segments ||--o| profiles : "updated\_by (SET NULL)"

segments ||--|{ segment\_additional\_terrain\_types : "has\_additional\_terrain (CASCADE)"

segment\_additional\_terrain\_types ||--|| terrain\_types\_master : "uses\_type (RESTRICT)"

segments ||--|{ segment\_media : "has\_gallery\_item (CASCADE)"

segment\_media ||--|| media : "links\_to\_media (CASCADE)"

segment\_media ||--o| media\_roles\_master : "has\_role (SET NULL)"

route\_segments ||--|| segments : "includes\_segment (CASCADE)" -- Context for RLS, actual link defined in route\_segments

routes ||--|{ route\_segments : "sequences\_segments" -- Context for RLS

trails ||--|{ routes : "contains\_route" -- Context for RLS

**6. Multilingual Strategy**

* **Translatable Fields:**
  + segments: name, short\_description, detailed\_description\_notes, waymarking\_on\_segment\_notes, segment\_suitability\_notes, water\_sources\_general\_notes, resupply\_options\_general\_notes, segment\_cultural\_historical\_notes, emergency\_access\_notes, segment\_weather\_advice.
  + segment\_media: caption, alt\_text.
* English base text stored directly. Other languages via public.translations table, linked by table\_name, column\_name, and row\_id (segments.id or segment\_media.id).

**7. Role-Based Workflow & RLS Notes**

* **Workflow Fields:** content\_visibility\_status for segments. Standard audit columns and deleted\_at in segments. Audit columns in junction tables segment\_additional\_terrain\_types and segment\_media.
* - Note: The RLS policies outlined above rely on the existence and correct implementation of global RLS helper functions (e.g., public.has\_role(TEXT), public.is\_platform\_admin(), specific regional/trail management checks) that authenticate users and verify their roles stored in the public.profiles table." This reinforces that the table-specific RLS is part of a larger auth system.
* **RLS Policy Stubs (Conceptual):**
  + **Public Read Access (**segments**,** segment\_additional\_terrain\_types**,** segment\_media**):** Granted if the segment is part of a published route on a published trail and deleted\_at IS NULL for the segment.
  + SQL

-- Policy for segments

CREATE POLICY "Allow public read access to published segments" ON public.segments FOR SELECT

USING (

segments.content\_visibility\_status = 'published' AND segments.deleted\_at IS NULL AND

EXISTS (SELECT 1 FROM public.route\_segments rs JOIN public.routes r ON rs.route\_id = r.id JOIN public.trails t ON r.trail\_id = t.id

WHERE rs.segment\_id = segments.id AND r.content\_visibility\_status = 'published' AND r.deleted\_at IS NULL AND t.content\_visibility\_status = 'published' AND t.deleted\_at IS NULL)

); --

-- Policy for segment\_additional\_terrain\_types

CREATE POLICY "Allow public read for segment\_additional\_terrain\_types of published segments" ON public.segment\_additional\_terrain\_types FOR SELECT

USING (EXISTS (SELECT 1 FROM public.segments s JOIN public.route\_segments rs ON rs.segment\_id = s.id JOIN public.routes r ON rs.route\_id = r.id JOIN public.trails t ON r.trail\_id = t.id WHERE s.id = segment\_additional\_terrain\_types.segment\_id AND s.content\_visibility\_status = 'published' AND s.deleted\_at IS NULL AND r.content\_visibility\_status = 'published' AND r.deleted\_at IS NULL AND t.content\_visibility\_status = 'published' AND t.deleted\_at IS NULL)); --

-- Policy for segment\_media

CREATE POLICY "Allow public read for segment\_media of published segments" ON public.segment\_media FOR SELECT

USING (EXISTS (SELECT 1 FROM public.segments s JOIN public.route\_segments rs ON rs.segment\_id = s.id JOIN public.routes r ON rs.route\_id = r.id JOIN public.trails t ON r.trail\_id = t.id WHERE s.id = segment\_media.segment\_id AND s.content\_visibility\_status = 'published' AND s.deleted\_at IS NULL AND r.content\_visibility\_status = 'published' AND r.deleted\_at IS NULL AND t.content\_visibility\_status = 'published' AND t.deleted\_at IS NULL)); --

* + **Admin Full Access (**segments **and junctions):** Full CRUD via helper like public.is\_platform\_admin().
  + **Regional Content Manager Access (**segments **and junctions):** Conditional access based on trail/region management via helper like public.is\_regional\_manager\_for\_segment(segment\_id).

**8. ENUM vs. Lookup Discussion**

* New ENUMs: public.segment\_sun\_exposure\_enum, public.segment\_travel\_direction\_enum are appropriate for limited, distinct values.
* dominant\_terrain\_type\_id correctly uses FK to public.terrain\_types\_master.
* other\_terrain\_types\_on\_segment (old TEXT[]) correctly replaced by public.segment\_additional\_terrain\_types junction table.

**9. UI/UX Enablement**

* **Filters:** segment\_difficulty, operational\_status, dominant\_terrain\_type\_id, etc.
* **Icons:** Driven by various status and type fields.
* **Core Data for Display:** path\_geom for maps; distance\_km, time, elevation stats for summaries; elevation\_profile\_data for charts. segment\_media for galleries.
* **🔴 Automated Geometric Calculations:** Essential for data accuracy and good UX.
* **API Support:** The public.segments\_summary\_view (defined in the Database Views Specification) can support efficient API responses for segment listings within a route.

**10. Key Considerations & Definitions**

* **🔴** path\_geom **as Source of Truth:** All derived geometric statistics must be calculated from it.
* Waypoint Linkage: start\_waypoint\_id <> end\_waypoint\_id enforced.
* **GPX Management:** Changed from gpx\_track\_data\_url (original spec) to gpx\_media\_id FK to public.media for V2, aligning with centralized media strategy.

**11. Scalability & Future-Proofing**

* BIGINT PK for segments.
* GIST index on path\_geom for spatial queries.
* Soft deletes and audit columns.
* Derived data automation.
* **Future Enhancement:** Full-Text Search (FTS) on descriptive fields is a Post-V2 consideration.
* **V2+ Consideration:** segments table is a candidate for partitioning.

**12. Next-Action Checklist**

1. 🔴 **Implement DDL:**
   * Create new ENUM types (public.segment\_sun\_exposure\_enum, public.segment\_travel\_direction\_enum).
   * Implement public.calculate\_segment\_geom\_properties() and public.update\_segment\_geom\_derived\_fields() functions.
   * Create public.segments table, replacing gpx\_track\_data\_url with gpx\_media\_id BIGINT REFERENCES public.media(id) ON DELETE SET NULL. Link triggers.
   * Create public.segment\_additional\_terrain\_types with its audit trigger.
   * Create public.segment\_media with media\_role\_code and alt\_text columns, and its audit trigger.
2. 🔴 **Implement Orphaned Translation Cleanup Triggers:**
   * Add an AFTER DELETE trigger on public.segments that calls public.cleanup\_related\_translations('segments', OLD.id).
   * Add an AFTER DELETE trigger on public.segment\_media that calls public.cleanup\_related\_translations('segment\_media', OLD.id) (for caption and alt\_text).
3. 🟠 **Confirm Referenced Table PK Data Types:** Verify public.waypoints.id and public.media.id are BIGINT; public.terrain\_types\_master.id is INTEGER; public.media\_roles\_master.code is TEXT.
4. 🟠 **Implement RLS Policies & Helper Functions:** Define and test RLS policies for segments, segment\_additional\_terrain\_types, and segment\_media.
5. 🟢 **Data Migration:** Plan migration for existing segment data, including mapping old fields to new structures (e.g., gpx\_track\_data\_url to gpx\_media\_id). Populate audit columns.
6. 🟢 **Populate** media\_roles\_master**:** Ensure relevant roles are defined.
7. 🟢 **Review API Usage:** Ensure API endpoints can efficiently query segment data, potentially using public.segments\_summary\_view.