<https://gemini.google.com/u/1/app/67787f5b9ee9db46>

<https://gemini.google.com/u/1/app/42a77d08d7a45bcb>

\* \* \* \* \*

Curated Itinerary Module Overview

Version: 2.1 (Checklist REV 05-18-25-A Applied, ENUM Promotion, Audit Columns, Translation Handling Finalized)

Date: May 18, 2025

This document provides a master recap and architecture overview for the Curated Itinerary Module (Module 7) of the pilgrimage-platform database. It details how tables, master data (formerly ENUMs), translation mechanisms, security policies, API considerations, and other database objects interlock, along with a recommended build order for successful deployment.

### 1\. Executive Summary

This database module provides the backbone for managing and presenting editorially curated pilgrimage itineraries. It allows administrators to define itineraries, their day-by-day structure (via segments), categorize them by themes and seasons, assign difficulty levels, and control their publication status using dedicated master tables. Key enhancements include the promotion of former ENUMs (difficulty, status, categories, seasons) to fully-fledged master lookup tables, the addition of comprehensive audit columns (`created\_at`, `updated\_at`, `created\_by\_profile\_id`, `updated\_by\_profile\_id`) to all tables, and standardized orphan-translation cleanup triggers. This module enables pilgrims to browse, filter, and select rich, localized journeys tailored to their preferences.

### 2\. Group-Level Snapshot

| Group | Key Tables | Primary Purpose | Top Inter-Group Links |

| 7\. Curated Itinerary Module | `curated\_itineraries`, `curated\_itinerary\_segments`, `itinerary\_categories\_master`, `curated\_itinerary\_to\_category`, `seasons\_master`, `curated\_itinerary\_to\_season`, `trail\_difficulty\_levels\_master`, `content\_statuses\_master` | Defines, structures, categorizes, and manages the lifecycle of pre-defined pilgrimage itineraries and their associated lookup data (difficulty, status, categories, seasons). | `profiles` (for audit columns), `media` (for itinerary images), `segments` (for itinerary composition), `public.translations` (for all multilingual content). |

\* \* \* \* \*

### 3\. Narrative Walkthrough

This module, "7. Curated Itinerary Module," is central to presenting rich, predefined pilgrimage experiences, now with enhanced structure and auditability.

- `trail\_difficulty\_levels\_master` (v2.1 - New Master Table):

- Stores the definitive, translatable list of difficulty levels (e.g., "Easy," "Moderate") with `code`, `name`, `description`, `icon\_identifier`, `sort\_order`, `is\_active` flag, and full V2 audit columns.

- Replaces the former `trail\_difficulty\_enum`.

- `AFTER DELETE` trigger cleans up related translations.

- `content\_statuses\_master` (v2.1 - New Master Table):

- Manages the translatable lifecycle statuses for content (e.g., "DRAFT," "PUBLISHED") with `code`, `name`, `description`, `icon\_identifier`, `sort\_order`, `is\_active` flag, and full V2 audit columns.

- Ensures content entities always have a defined status via a FK relationship.

- Replaces the former `content\_visibility\_status\_enum`.

- `AFTER DELETE` trigger cleans up related translations.

- `itinerary\_categories\_master` (v2.1 - New Master Table):

- A lookup table for translatable itinerary categories (e.g., "Franciscan Sites Focus") with `category\_code`, `name`, `description`, `icon\_identifier`, `sort\_order`, `is\_active` flag, and full V2 audit columns.

- Replaces the former `itinerary\_category\_tag\_enum[]`.

- `AFTER DELETE` trigger cleans up related translations.

- `seasons\_master` (v2.1 - New Master Table):

- A lookup table for translatable seasons (e.g., "Spring," "Year-round") with `season\_code`, `name`, `description`, `icon\_identifier`, `sort\_order`, `is\_active` flag, and full V2 audit columns.

- Replaces former `TEXT[]` for seasons.

- `AFTER DELETE` trigger cleans up related translations.

- `curated\_itineraries` (v3.1):

- Core table holding metadata for each curated journey. Includes translatable fields like `name`, `short\_description`, `long\_description`, etc.

- Links to `trail\_difficulty\_levels\_master` via `overall\_difficulty\_level\_id` (FK).

- Links to `content\_statuses\_master` via `content\_status\_id` (FK, NOT NULL).

- Includes full V2 audit columns (`created\_at`, `updated\_at`, `created\_by\_profile\_id`, `updated\_by\_profile\_id`) and `deleted\_at` for soft deletion.

- Comprehensive audit trigger and orphan-translation cleanup trigger are implemented.

- Default `content\_status\_id` (to 'DRAFT') and `slug` generation are application/trigger responsibilities.

- `curated\_itinerary\_to\_category` (v2.1 - New Junction Table):

- Establishes M-M relationship between `curated\_itineraries` and `itinerary\_categories\_master`.

- Includes `itinerary\_id` (CASCADE DELETE), `category\_id` (RESTRICT DELETE), and full V2 audit columns.

- Comprehensive audit trigger implemented.

- `curated\_itinerary\_to\_season` (v2.1 - New Junction Table):

- Establishes M-M relationship between `curated\_itineraries` and `seasons\_master`.

- Includes `itinerary\_id` (CASCADE DELETE), `season\_id` (RESTRICT DELETE), and full V2 audit columns.

- Comprehensive audit trigger implemented.

- `curated\_itinerary\_segments` (v2.1):

- Defines the ordered "playlist" of `segments` for each day of an itinerary.

- Includes `itinerary\_id` (CASCADE DELETE), `segment\_id` (RESTRICT DELETE), `day\_number\_in\_itinerary`, `order\_of\_segment\_within\_day`.

- `notes\_for\_segment\_in\_itinerary` is translatable.

- Includes full V2 audit columns.

- Comprehensive audit trigger and orphan-translation cleanup trigger implemented.

- `UNIQUE (itinerary\_id, day\_number\_in\_itinerary, order\_of\_segment\_within\_day)` constraint.

\* \* \* \* \*

### 4\. Cross-Cutting Concerns

- Users & Roles:

- All tables in this module now include standardized V2 audit columns: `created\_at`, `updated\_at`, `created\_by\_profile\_id` (FK to `profiles.id`), and `updated\_by\_profile\_id` (FK to `profiles.id`).

- Comprehensive audit triggers populate these fields.

- RLS policies differentiate access based on roles, utilizing helper functions.

- Translations / i18n:

- A central `public.translations` table stores all multilingual content.

- Base language (English) versions are stored directly in respective table columns (e.g., `curated\_itineraries.name`, `itinerary\_categories\_master.description`). Column names are now generic (e.g., `name`, not `name\_en`).

- Translatable Fields Mapping (Module 7 - V2.1):

- `curated\_itineraries`: `name`, `short\_description`, `long\_description`, `theme\_or\_focus`, `suitability\_notes`, `primary\_start\_location\_text`, `primary\_end\_location\_text`.

- `curated\_itinerary\_segments`: `notes\_for\_segment\_in\_itinerary`.

- `itinerary\_categories\_master`: `name`, `description`.

- `seasons\_master`: `name`, `description`.

- `trail\_difficulty\_levels\_master`: `name`, `description`.

- `content\_statuses\_master`: `name`, `description`.

- 🔴 Crucial: All tables with translatable fields (`curated\_itineraries`, `curated\_itinerary\_segments`, and all four new master tables) now have `AFTER DELETE` triggers that call specific cleanup functions (e.g., `public.cleanup\_curated\_itineraries\_translations()`) to remove orphaned entries from `public.translations`.

- ENUM & Taxonomy Registry (V2.1 Update: Promotion to Master Tables):

- `curated\_itineraries.overall\_difficulty\_level\_id` (FK) now points to `public.trail\_difficulty\_levels\_master(id)`.

- `curated\_itineraries.content\_status\_id` (FK) now points to `public.content\_statuses\_master(id)`.

- Itinerary categories are managed via `public.itinerary\_categories\_master` and linked through `public.curated\_itinerary\_to\_category`.

- Recommended seasons are managed via `public.seasons\_master` and linked through `public.curated\_itinerary\_to\_season`.

- All new master tables include `code`, `name` (translatable), `description` (translatable), `icon\_identifier` (optional), `sort\_order`, `is\_active`, and full audit columns.

- Media & Files:

- `curated\_itineraries` links to `public.media(id)` (UUID PK) for `banner\_image\_media\_id` and `map\_overview\_image\_media\_id` using `uuid` FKs.

- `ON DELETE SET NULL` is used for these media FKs.

- Audit / Soft-Delete / Versioning:

- Audit: All tables feature full V2 audit columns, managed by triggers.

- Soft-Delete: `curated\_itineraries` uses `deleted\_at timestamp with time zone`. Master lookup tables use an `is\_active boolean` flag. Junction tables and `curated\_itinerary\_segments` do not have soft delete; their lifecycle is tied to parent records.

- Versioning: `curated\_itineraries` has `version\_notes text`. `updated\_at` provides basic versioning hooks.

\* \* \* \* \*

### 5\. Security & Access Control 🔐

- RLS Overview:

- Row-Level Security policies are defined for all tables to control access based on user roles (e.g., Admin, Regional Content Manager, Authenticated User, Anonymous) and content status (e.g., 'PUBLISHED', `is\_active=true`).

- Policies typically grant full CUD to Admins. Content Managers have conditional write access. Public users have read access to published/active content.

- RLS Helper Functions: Policies rely on helper functions like `public.has\_role(TEXT)`, `public.is\_platform\_admin()`, and potentially module-specific helpers (e.g., `user\_manages\_itinerary\_content()`). These functions query `public.profiles.roles` using `auth.uid()`.

- SECURITY DEFINER Functions: Audit triggers and translation cleanup triggers are `SECURITY DEFINER` and must have their `search\_path` appropriately set (e.g., `ALTER FUNCTION ... SET search\_path = public;`).

\* \* \* \* \*

### 6\. Prerequisite Objects & Build Order ⚙️

1. Global/Shared Prerequisite Objects:

- `public.profiles` table (for audit FKs).

- `public.media` table (for image FKs).

- `public.segments` table (for `curated\_itinerary\_segments.segment\_id` FK).

- `public.translations` table (for storing all translations).

- Core RLS helper functions (`public.has\_role(TEXT)`, `public.is\_platform\_admin()`).

- Standard audit trigger function (e.g., `public.set\_master\_table\_audit\_meta()` or `public.set\_linking\_table\_audit\_meta()`).

- Standard translation cleanup function template adapted for each table.

2. Module 7 Tables & Objects (Recommended Build Order):

1. Master Tables (Lookup Data):

- `public.trail\_difficulty\_levels\_master` (DDL, Triggers: Audit, Translations Cleanup)

- `public.content\_statuses\_master` (DDL, Triggers: Audit, Translations Cleanup)

- `public.itinerary\_categories\_master` (DDL, Triggers: Audit, Translations Cleanup)

- `public.seasons\_master` (DDL, Triggers: Audit, Translations Cleanup)

2. Seed Master Tables: Populate the above master tables with initial data and their base language translations in `public.translations`.

3. Core Entity Table:

- `public.curated\_itineraries` (DDL including FKs to master tables, Triggers: Audit, Translations Cleanup)

4. Linking/Detail Tables:

- `public.curated\_itinerary\_to\_category` (DDL, Triggers: Audit)

- `public.curated\_itinerary\_to\_season` (DDL, Triggers: Audit)

- `public.curated\_itinerary\_segments` (DDL, Triggers: Audit, Translations Cleanup)

5. Views (for API support):

- `public.v\_curated\_itineraries\_list\_localized`

- `public.v\_curated\_itinerary\_detail\_localized`

- \*(Consider `public.v\_curated\_itinerary\_segments\_localized`)\*

6. Indexes & Constraints: Apply all remaining indexes and constraints not created with table DDLs.

7. RLS Policies: Enable RLS and apply policies to all tables.

\* \* \* \* \*

### 7\. Performance & Optimization Extras

- Key Indexes: Comprehensive indexing is defined for all tables, including on PKs, FKs, unique constraints (`code`, `name`, `slug`), boolean flags (`is\_active`, `is\_featured\_itinerary`), and `sort\_order` on master tables. The `translations` table requires a robust composite index.

- Views for API Performance:

- `public.v\_curated\_itineraries\_list\_localized`: Designed for efficient listing of itineraries with key localized information.

- `public.v\_curated\_itinerary\_detail\_localized`: Provides comprehensive localized details for a single itinerary, using CTEs for JSONB aggregation of categories and seasons. Performance should be monitored, and materialization or DB functions considered if necessary.

- Partitioning: Not anticipated as necessary for Module 7 tables in V2.1.

- Caching: Application-level caching for frequently accessed, relatively static data (like master table content or published itinerary details) is recommended.

\* \* \* \* \*

### 8\. Visuals (Conceptual ERD - Module 7 Focus)

Code snippet

```

erDiagram

profiles {

uuid id PK

text public\_display\_name

}

media {

uuid id PK

text storage\_object\_path\_original

jsonb image\_variants\_json

text default\_alt\_text "Translatable"

}

segments {

bigint id PK

text name "Translatable"

}

translations {

bigint id PK

text table\_identifier

text column\_identifier

text row\_foreign\_key

text language\_code FK

text translated\_text

}

trail\_difficulty\_levels\_master {

integer id PK

text code UK

text name "Base Lang, Translatable"

text description "Base Lang, Translatable"

text icon\_identifier

integer sort\_order

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

content\_statuses\_master {

integer id PK

text code UK

text name "Base Lang, Translatable"

text description "Base Lang, Translatable"

text icon\_identifier

integer sort\_order

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

itinerary\_categories\_master {

integer id PK

text category\_code UK

text name "Base Lang, Translatable"

text description "Base Lang, Translatable"

text icon\_identifier

integer sort\_order

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

seasons\_master {

integer id PK

text season\_code UK

text name "Base Lang, Translatable"

text description "Base Lang, Translatable"

text icon\_identifier

integer sort\_order

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

curated\_itineraries {

bigint id PK

text name "Base Lang, Translatable"

text slug UK

text short\_description "Base Lang, Translatable"

text long\_description "Base Lang, Translatable"

integer overall\_difficulty\_level\_id FK

integer content\_status\_id FK

uuid banner\_image\_media\_id FK

uuid map\_overview\_image\_media\_id FK

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

timestamptz deleted\_at

}

curated\_itinerary\_to\_category {

bigint itinerary\_id PK FK

integer category\_id PK FK

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

curated\_itinerary\_to\_season {

bigint itinerary\_id PK FK

integer season\_id PK FK

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

curated\_itinerary\_segments {

bigint id PK

bigint itinerary\_id FK

bigint segment\_id FK

text notes\_for\_segment\_in\_itinerary "Base Lang, Translatable"

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

curated\_itineraries }o--|| trail\_difficulty\_levels\_master : "difficulty (SET NULL)"

curated\_itineraries }o--|| content\_statuses\_master : "status (RESTRICT)"

curated\_itineraries }o--|| media : "banner\_image (SET NULL)"

curated\_itineraries }o--|| media : "map\_image (SET NULL)"

curated\_itineraries }o--|| profiles : "created\_by (SET NULL)"

curated\_itineraries }o--|| profiles : "updated\_by (SET NULL)"

curated\_itineraries ||--|{ curated\_itinerary\_to\_category : "has\_categories (CASCADE)"

itinerary\_categories\_master ||--|{ curated\_itinerary\_to\_category : "categorizes (RESTRICT)"

curated\_itinerary\_to\_category }o--|| profiles : "link\_created\_by (SET NULL)"

curated\_itinerary\_to\_category }o--|| profiles : "link\_updated\_by (SET NULL)"

curated\_itineraries ||--|{ curated\_itinerary\_to\_season : "has\_seasons (CASCADE)"

seasons\_master ||--|{ curated\_itinerary\_to\_season : "applies\_to (RESTRICT)"

curated\_itinerary\_to\_season }o--|| profiles : "link\_created\_by (SET NULL)"

curated\_itinerary\_to\_season }o--|| profiles : "link\_updated\_by (SET NULL)"

curated\_itineraries ||--|{ curated\_itinerary\_segments : "composed\_of (CASCADE)"

curated\_itinerary\_segments }o--|| segments : "uses\_segment (RESTRICT)"

curated\_itinerary\_segments }o--|| profiles : "segment\_link\_created\_by (SET NULL)"

curated\_itinerary\_segments }o--|| profiles : "segment\_link\_updated\_by (SET NULL)"

trail\_difficulty\_levels\_master }o--|| profiles : "master\_created\_by (SET NULL)"

trail\_difficulty\_levels\_master }o--|| profiles : "master\_updated\_by (SET NULL)"

content\_statuses\_master }o--|| profiles : "master\_created\_by (SET NULL)"

content\_statuses\_master }o--|| profiles : "master\_updated\_by (SET NULL)"

itinerary\_categories\_master }o--|| profiles : "master\_created\_by (SET NULL)"

itinerary\_categories\_master }o--|| profiles : "master\_updated\_by (SET NULL)"

seasons\_master }o--|| profiles : "master\_created\_by (SET NULL)"

seasons\_master }o--|| profiles : "master\_updated\_by (SET NULL)"

%% Conceptual links to translations table for translatable fields in each relevant table

curated\_itineraries ..> translations : "translatable\_fields"

itinerary\_categories\_master ..> translations : "translatable\_fields"

seasons\_master ..> translations : "translatable\_fields"

trail\_difficulty\_levels\_master ..> translations : "translatable\_fields"

content\_statuses\_master ..> translations : "translatable\_fields"

curated\_itinerary\_segments ..> translations : "translatable\_fields"

```

\* \* \* \* \*

### 9\. Critical Gaps & Risks (Post V2.1 Review)

- 🟠 Default `content\_status\_id` and `slug` Generation for `curated\_itineraries`: Requires robust implementation (application logic or DB trigger). If inconsistent, itineraries might be unusable or have poor SEO.

- 🟠 Performance of Detailed Views: `v\_curated\_itinerary\_detail\_localized` with its multiple joins and JSONB aggregations needs thorough performance testing under load. Consider materialization or DB functions if it becomes a bottleneck.

- 🟢 RLS Helper Function Implementation: Assumed helper functions (`is\_platform\_admin()`, etc.) must be securely and correctly implemented.

- 🟢 Seed Data Accuracy: Initial values for master tables require business/content team validation.

- 🟢 Application Layer Logic for Translations: The DB provides the mechanism, but the application layer must correctly request localized content, manage translations for new/updated base language text, and handle fallbacks if a translation is missing.

\* \* \* \* \*

### 10\. Scalability & Future-Proof Notes

- Promotion to Master Tables: Using dedicated master tables for difficulty, status, categories, and seasons provides excellent scalability and flexibility for adding new attributes or options without schema changes to `curated\_itineraries`.

- Standardized Auditing & i18n: Full audit trails and a central translation mechanism with orphan cleanup improve data integrity and maintainability.

- Surrogate PKs on Junctions (if applicable): `curated\_itinerary\_segments` uses a surrogate PK, which can be beneficial if these links ever need their own direct translations or more complex attributes. `curated\_itinerary\_to\_category` and `curated\_itinerary\_to\_season` use composite PKs, suitable for pure M-M links.

- Views for API Decoupling: Using views helps decouple the API from the physical schema.

\* \* \* \* \*

### 11\. Next Steps (Module 7 Deployment)

1. P1: Implement Prerequisite Global Objects: Ensure `public.profiles`, `public.media`, `public.segments`, `public.translations` tables, RLS helper functions, and standard audit trigger functions are in place.

2. P1: Execute DDL for Module 7: Create all master tables, then `curated\_itineraries`, then junction/detail tables, including all indexes and triggers (audit and translation cleanup).

3. P1: Seed Master Tables: Populate `trail\_difficulty\_levels\_master`, `content\_statuses\_master`, `itinerary\_categories\_master`, `seasons\_master` with initial data and their base language translations.

4. P1: Implement and Test RLS Policies: Apply and rigorously test all RLS policies for all Module 7 tables.

5. P2: Implement Views: Create and test `public.v\_curated\_itineraries\_list\_localized` and `public.v\_curated\_itinerary\_detail\_localized`.

6. P2: Develop/Update Application Logic: Adapt backend services and admin UIs to use the new V2.1 structures, manage translations, and interact with the new master tables and views.

7. P2: Data Migration (If Applicable): Migrate any existing itinerary data to the new V2.1 schema, including populating new FKs and audit columns.

\* \* \* \* \*