<https://gemini.google.com/u/1/app/cd801fb8de131fa6>

<https://gemini.google.com/u/1/app/0e3ccfac5f6b71bb>

### Updated Module Overview: 4d. Waypoint - Events

Version: 1.1 (reflecting V2.1 checklist application and i18n corrections)

Date: May 18, 2025

This document provides a master recap and architecture overview for the "4d. Waypoint - Events" database module of the pilgrimage-platform. It details how tables, 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 a robust framework for managing detailed information about events associated with pilgrimage waypoints. It enables pilgrims to discover and plan around local festivals, markets, and other happenings, while empowering content managers to curate and maintain accurate event data using standardized classifications and audit trails, thereby enriching the overall pilgrimage experience. All translatable text fields store their primary reference language content directly, with other language versions managed in the central `public.translations` table.

#### 2\. Group-Level Snapshot

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

| 4d. Waypoint - Events | `events\_details` (v1.3), `event\_types\_master` (v1.1), `event\_recurrence\_frequencies\_master` (v1.1), `event\_attendance\_scales\_master` (v1.1), `event\_date\_certainty\_levels\_master` (v1.1) | Stores comprehensive details for events linked to waypoints, including their classification, timing, scale, and data verification. | `waypoints` (Module 4a - Core Waypoint Info), `profiles` (Module 1 - User Info), `tags\_master` (Module 4a - Shared Tags), `public.translations` (Module 1) |

\* \* \* \* \*

#### 3\. Narrative Walkthrough

The "4d. Waypoint - Events" group centralizes all event-specific information.

- `event\_types\_master` (v1.1): Defines the various categories of events (e.g., "Religious Feast Day," "Local Market"). Its primary key is an auto-generated `id` (INTEGER). It includes `code`, `default\_name`, `default\_description`, `icon\_identifier`, `sort\_order`, `is\_active`, and standard audit columns.

- `event\_recurrence\_frequencies\_master` (v1.1): Standardizes terms for how often recurring events occur (e.g., "Annually," "Weekly"). Its primary key is an auto-generated `id` (INTEGER). It includes `code`, `default\_name`, `default\_description`, `sort\_order`, `is\_active`, and standard audit columns.

- `event\_attendance\_scales\_master` (v1.1): Categorizes events by expected attendance size (e.g., "Small - Local," "Large - National"). Its primary key is an auto-generated `id` (INTEGER). It includes `code`, `default\_name`, `default\_description`, `approximate\_range`, `sort\_order`, `is\_active`, and standard audit columns.

- `event\_date\_certainty\_levels\_master` (v1.1): Specifies the confidence level for future dates of recurring events (e.g., "Confirmed," "Pattern-Based Estimate"). Its primary key is an auto-generated `id` (INTEGER). It includes `code`, `default\_name`, `default\_description`, `default\_advice`, `sort\_order`, `is\_active`, and standard audit columns.

- `events\_details` (v1.3): This is the central table, storing detailed information for waypoints that are event locations. Its primary key is `waypoint\_id` (BIGINT), which is also a foreign key to `public.waypoints(id)`.

- Key Relationships & Cardinalities:

- It has a 1-to-1 relationship with the `waypoints` table (via `waypoint\_id`).

- Connects to `event\_types\_master` via `event\_type\_id` (Many-to-One).

- Connects to `event\_recurrence\_frequencies\_master` via `recurrence\_frequency\_id` (Many-to-One).

- Connects to `event\_attendance\_scales\_master` via `expected\_attendance\_scale\_id` (Many-to-One).

- Connects to `event\_date\_certainty\_levels\_master` via `future\_date\_estimation\_level\_id` (Many-to-One).

- Connects to `tags\_master` via `event\_theme\_or\_focus\_tag\_ids` (INTEGER[]), representing a Many-to-Many relationship for event themes.

- Critical Triggers:

- `trigger\_validate\_event\_theme\_tags` (BEFORE INSERT OR UPDATE on `events\_details`) ensures `event\_theme\_or\_focus\_tag\_ids` reference valid and active entries in `tags\_master`.

- All tables in this module have a trigger (e.g., `handle\_\*\_updated\_at`) to automatically update the `updated\_at` timestamp on row modification.

- All tables in this module (`events\_details` and the four master tables) have an `AFTER DELETE` trigger (e.g., `trigger\_cleanup\_\*\_translations`) to remove their associated orphaned entries from the `public.translations` table.

- Default Values:

- `events\_details`: `is\_recurring\_event` defaults to `false`; `created\_at` and `updated\_at` default to `now()`.

- Master tables: `id` is `GENERATED ALWAYS AS IDENTITY`, `sort\_order` defaults to `0`, `is\_active` defaults to `true`, and `created\_at`, `updated\_at` default to `now()`.

\* \* \* \* \*

#### 4\. Cross-Cutting Concerns

- Users & Roles:

- Ownership and moderation are tracked in `events\_details` via `created\_by\_profile\_id`, `updated\_by\_profile\_id`, and `data\_verified\_by\_profile\_id`, which are foreign keys to `public.profiles(id)`.

- All master tables in this module also now include `created\_by\_profile\_id` and `updated\_by\_profile\_id`.

- RLS policies leverage these, typically granting broader access based on user roles (e.g., 'admin\_platform', 'regional\_content\_manager') using helper functions like `public.has\_role\_on\_profile()`.

- Translations / i18n:

- Numerous text fields in `events\_details` (e.g., `default\_event\_name\_official`, `default\_description\_long`) store their primary reference language content directly and are marked as translatable via the central `public.translations` table.

- For the master tables, fields like `default\_name`, `default\_description` (and `default\_advice` for `event\_date\_certainty\_levels\_master`; `approximate\_range` for `event\_attendance\_scales\_master`) store primary reference language text and are translatable via `public.translations`.

- `event\_theme\_or\_focus\_tag\_ids` links to `tags\_master`, whose `default\_name` field is assumed to be translated via `public.translations`.

- AFTER DELETE triggers on all tables in this module ensure orphaned translations are cleaned up.

- ENUM & Taxonomy Registry:

- All previous ENUM types related to events have been promoted to dedicated master lookup tables within this module, each with `code`, `default\_name`, `is\_active`, `sort\_order`, and audit columns.

- The text array `event\_theme\_or\_focus\_tags` in `events\_details` was correctly replaced by `event\_theme\_or\_focus\_tag\_ids` (integer array), linking to `tags\_master(id)`.

- Media & Files:

- The `event\_types\_master` table includes an `icon\_identifier` column (TEXT) for UI icons. No other tables in this module directly link to `public.media`. Event-specific images would typically be associated via the parent `waypoints` media linking strategy (e.g., `waypoint\_media`).

- Audit / Soft-Delete / Versioning:

- Standard Audit Columns: All tables in this module (`events\_details` and the four master tables) now include `created\_at`, `updated\_at` (auto-updated), `created\_by\_profile\_id` (FK to `profiles`), and `updated\_by\_profile\_id` (FK to `profiles`).

- `events\_details` has specific data verification tracking with `data\_last\_verified\_at` and `data\_verified\_by\_profile\_id`.

- Soft-Delete: `events\_details` includes a `deleted\_at` timestamp column. Master tables use an `is\_active BOOLEAN NOT NULL DEFAULT true` flag for lifecycle management.

- Versioning: Schema versions are noted in table comments. Content versioning is not implemented at the row level.

\* \* \* \* \*

#### 5\. Security & Access Control 🔐

- RLS Overview: Row-Level Security policies are implemented for all tables in this module.

- Public users generally have read-only access to active master data and event details linked to published and non-deleted waypoints.

- Authenticated users with specific roles (e.g., 'admin\_platform', 'regional\_content\_manager', or a dedicated 'event\_manager' if defined) have broader CRUD permissions, often contingent on the parent waypoint's status or regional assignments.

- Policies extensively use helper functions like `public.has\_role\_on\_profile(auth.uid(), 'role\_code')` and `auth.uid()` for ownership checks.

- Dedicated `SECURITY DEFINER` functions:

- `public.check\_event\_theme\_tags\_exist()`: Validates tags linked to `events\_details`.

- `public.cleanup\_\*\_translations()`: Functions for cleaning up orphaned translations for each table in this module. These must have their `search\_path` hardened.

- Anon vs Authenticated Access: Anonymous users have read-only access to published event information and active master list data. Authenticated users, depending on their roles, gain write/update capabilities.

\* \* \* \* \*

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

The following order should be observed for deployment (assuming base types, `public.profiles`, `public.waypoints`, `public.tags\_master`, `public.translations`, and helper function `extensions.moddatetime` or equivalent already exist):

1. Functions (SQL/PLPGSQL):

- `public.check\_event\_theme\_tags\_exist()` (updated to check `tags\_master.is\_active`).

- `public.cleanup\_event\_types\_master\_translations()`.

- `public.cleanup\_event\_recurrence\_frequencies\_master\_translations()`.

- `public.cleanup\_event\_attendance\_scales\_master\_translations()`.

- `public.cleanup\_event\_date\_certainty\_levels\_master\_translations()`.

- `public.cleanup\_events\_details\_translations()`.

2. Core Tables (with PK/FK, audit columns, `is\_active` flags):

- `public.event\_types\_master` (v1.1)

- `public.event\_recurrence\_frequencies\_master` (v1.1)

- `public.event\_attendance\_scales\_master` (v1.1)

- `public.event\_date\_certainty\_levels\_master` (v1.1)

- `public.events\_details` (v1.3)

3. Seed Data: Populate all master tables with initial seed data (including `default\_name`, `code`, `is\_active`, `sort\_order`, and placeholder audit user IDs).

4. Views:

- `public.event\_details\_enriched\_view` (optional but recommended for API efficiency).

5. Indexes: Apply all defined indexes on these tables (for FKs, `is\_active`/`sort\_order` pairs, GIN indexes, etc.).

6. Triggers:

- Apply `updated\_at` triggers to all five tables.

- Apply `trigger\_validate\_event\_theme\_tags` to `events\_details`.

- Apply orphan translation cleanup triggers (`trigger\_cleanup\_\*\_translations`) to all five tables.

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

\* \* \* \* \*

#### 7\. API Endpoints Summary (Conceptual)

- `GET /events`: Lists event details, supporting filters (date, type, location), pagination, and language preferences. Leverages the `event\_details\_enriched\_view`.

- `POST /events`: Creates a new event detail record (linked to a waypoint).

- `GET /events/{waypoint\_id}`: Retrieves comprehensive details for a specific event. Leverages the `event\_details\_enriched\_view`.

- `PATCH /events/{waypoint\_id}`: Updates an existing event detail record.

- `GET /event\_types`: Lists available event types.

- `GET /event\_recurrence\_frequencies`: Lists available recurrence frequencies.

- `GET /event\_attendance\_scales`: Lists available attendance scales.

- `GET /event\_date\_certainty\_levels`: Lists available date certainty levels.

\* \* \* \* \*

#### 8\. Performance & Optimization Extras

- Key Indexes:

- `events\_details`: on `waypoint\_id` (PK), `event\_type\_id`, `start\_datetime`, `is\_recurring\_event`, GIN index on `event\_theme\_or\_focus\_tag\_ids`, and `deleted\_at`. Indexes on audit FKs and `related\_attraction\_waypoint\_id`.

- Master tables: PK on `id`, UNIQUE on `code`, and on `(is\_active, sort\_order)`. Indexes on audit FKs.

- View Usage: The `public.event\_details\_enriched\_view` is recommended to optimize read operations by pre-joining data.

- Partitioning: Not planned for V1. `events\_details` could be a candidate for date-based partitioning in the future if volume grows significantly.

- Caching: Master data tables are good candidates for application-level caching due to their relatively static nature.

\* \* \* \* \*

#### 9\. Visuals (Conceptual ERD - Module Focus)

Code snippet

```

erDiagram

waypoints {

bigint id PK

text default\_name

uuid content\_visibility\_status\_id FK

boolean deleted\_at

}

profiles {

uuid id PK

text public\_display\_name

}

tags\_master {

integer id PK

text default\_name

boolean is\_active

}

translations {

bigint id PK

text table\_identifier

text column\_identifier

text row\_foreign\_key

text language\_code FK

}

event\_types\_master {

integer id PK

text code UK

text default\_name

text icon\_identifier

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

event\_recurrence\_frequencies\_master {

integer id PK

text code UK

text default\_name

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

event\_attendance\_scales\_master {

integer id PK

text code UK

text default\_name

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

event\_date\_certainty\_levels\_master {

integer id PK

text code UK

text default\_name

boolean is\_active

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

}

events\_details {

bigint waypoint\_id PK FK

text default\_event\_name\_official

integer event\_type\_id FK

integer[] event\_theme\_or\_focus\_tag\_ids FK

timestamptz start\_datetime

boolean is\_recurring\_event

integer recurrence\_frequency\_id FK

integer future\_date\_estimation\_level\_id FK

integer expected\_attendance\_scale\_id FK

uuid created\_by\_profile\_id FK

uuid updated\_by\_profile\_id FK

uuid data\_verified\_by\_profile\_id FK

timestamptz deleted\_at

}

events\_details ||--o{ waypoints : "details\_for\_waypoint"

events\_details }o--|| event\_types\_master : "is\_of\_type"

events\_details }o--o| event\_recurrence\_frequencies\_master : "recurs\_at\_frequency"

events\_details }o--o| event\_attendance\_scales\_master : "has\_attendance\_scale"

events\_details }o--o| event\_date\_certainty\_levels\_master : "has\_date\_certainty"

events\_details }o--o| tags\_master : "themed\_with\_tags (array)"

events\_details }o--o| profiles : "created\_by"

events\_details }o--o| profiles : "updated\_by"

events\_details }o--o| profiles : "data\_verified\_by"

event\_types\_master o--o{ profiles : "audit\_info"

event\_recurrence\_frequencies\_master o--o{ profiles : "audit\_info"

event\_attendance\_scales\_master o--o{ profiles : "audit\_info"

event\_date\_certainty\_levels\_master o--o{ profiles : "audit\_info"

event\_types\_master ..> translations : "translatable\_fields"

event\_recurrence\_frequencies\_master ..> translations : "translatable\_fields"

event\_attendance\_scales\_master ..> translations : "translatable\_fields"

event\_date\_certainty\_levels\_master ..> translations : "translatable\_fields"

events\_details ..> translations : "translatable\_fields"

```

\* \* \* \* \*

#### 10\. Data & Workflow Flowchart (Simplified)

1. Admin/Content Manager (Setup & Maintenance):

- Populates/Manages master tables (`event\_types\_master`, `event\_recurrence\_frequencies\_master`, etc.) ensuring `is\_active` is true for usable entries. Audit fields (`created\_by\_profile\_id`, `updated\_by\_profile\_id`) are populated.

2. Admin/Content Manager (Event Creation/Update):

- Selects or creates a parent `waypoint`.

- Creates/updates an `events\_details` record, linking it to the `waypoint\_id`.

- Fills in `default\_event\_name\_official`, `start\_datetime`, selects type, frequency, scale, certainty from master tables, and links tags.

- `check\_event\_theme\_tags\_exist` trigger validates tags. `updated\_at` trigger fires. Audit fields populated.

3. Translator (Content Internationalization):

- (Via separate interface/process) Adds translations for `default\_` fields from `events\_details` and master tables into `public.translations`, linking via `table\_identifier`, `column\_identifier`, and `row\_foreign\_key`.

4. Admin/Content Manager (Verification):

- Periodically verifies event data, updating `data\_last\_verified\_at` and `data\_verified\_by\_profile\_id` in `events\_details`.

5. Pilgrim/End-User (Consumption):

- Accesses event data via API (e.g., `GET /events` or `GET /events/{waypoint\_id}`).

- API layer (potentially using `event\_details\_enriched\_view` and RPC functions for translations) fetches event details.

- RLS policies ensure only published/accessible events are shown.

- Application UI interprets date logic and displays event information in user's preferred language.

6. System (Deletion Cleanup):

- If an `events\_details` record (or a master table record) is deleted, its corresponding `AFTER DELETE` trigger fires to remove associated entries from `public.translations`.

\* \* \* \* \*

#### 11\. Critical Gaps & Risks

- 🔴 `tags\_master.is\_active` Dependency: The `trigger\_validate\_event\_theme\_tags` on `events\_details` requires `tags\_master` to have an `is\_active` column for correct validation. This external dependency must be met. (User confirmed `tags\_master` is updated).

- 🟠 Application Logic for Date Interpretation: The complexity of accurately interpreting and displaying recurring event dates (considering `is\_recurring\_event`, `recurrence\_frequency\_id`, `default\_recurrence\_detail\_text`, `future\_date\_estimation\_level\_id`, `exception\_dates`, `default\_specific\_next\_occurrence\_dates\_text`) resides heavily in the application layer and requires robust implementation and thorough testing.

- 🟠 Translation Workflow: A clear workflow and interface for managing translations in `public.translations` for all translatable fields in this module is essential.

- 🟢 Iconography Management: A system for managing and associating `icon\_identifier` values from `event\_types\_master` with actual UI assets needs to be in place.

\* \* \* \* \*

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

- Structured Data: The use of master tables and detailed, specific columns in `events\_details` provides a strong, flexible foundation for current and future needs.

- Auditability: Comprehensive audit trails (`created\_at`, `updated\_at`, `created\_by\_profile\_id`, `updated\_by\_profile\_id` on all tables, plus specific verification fields in `events\_details`) enhance data governance.

- Internationalization: Centralized translation strategy is robust.

- Future Date Queries: For very complex querying based on calculated future occurrences of recurring events, a dedicated `event\_occurrences` table might be a future (V2+) consideration if performance with current structure becomes an issue for such specific queries.

\* \* \* \* \*

#### 13\. Next Steps (Module Completion & Integration)

1. P1: Final DDL Execution: Execute all DDL for tables, functions, triggers, indexes, and views as finalized for this module, respecting the build order.

2. P1: Seed Master Data: Populate all master tables (`event\_types\_master`, `event\_recurrence\_frequencies\_master`, `event\_attendance\_scales\_master`, `event\_date\_certainty\_levels\_master`) with agreed initial data, ensuring audit columns are appropriately set.

3. P1: Implement RLS Policies: Apply all Row-Level Security policies.

4. P2: Develop API Layer: Implement API endpoints, leveraging the `event\_details\_enriched\_view` and creating necessary RPC functions for data fetching and translation assembly.

5. P2: Application Logic: Develop robust application-side logic for date interpretation, display, and filtering.

6. P2: Content Management Interface: Ensure the admin/content management UI correctly supports all fields, including links to master data and the new audit/verification fields.

7. P2: Testing: Conduct thorough testing of data integrity (triggers, constraints), API functionality, RLS, and the date logic.