Training System Specification

Overview

The training system is built on a hierarchical structure that organizes climbing training from high-level plans down to individual exercises. This hierarchy - Plan → Block → Workout → Exercise - provides a flexible framework that can accommodate different training philosophies while maintaining consistent organization.

Each level in the hierarchy serves a specific purpose in organizing training, with clear relationships between levels. Higher levels provide structure and progression strategy, while lower levels handle the practical details of individual training sessions.

1. Training Plan

A training plan represents the highest level of organization in the system. Plans coordinate multiple training blocks according to a defined progression strategy, either linear (blocks follow in strict sequence) or nonlinear (blocks rotate or alternate based on specific patterns).

The core purpose of a plan is to provide structure to training blocks while maintaining flexibility in implementation. Plans can be either fixed-duration with specific end dates or open-ended, allowing for different approaches to long-term training organization.

Required Attributes:

- Unique identifier
- Name
- Progression type: linear | nonlinear
- Start date

- Training blocks (ordered list)

Optional Attributes:

- End date (required for fixed-duration plans)
- Target weekly hours

Progression Rules:
Plans support two fundamental progression types, each with distinct behavior:

Linear progression maintains blocks in a fixed sequence, appropriate for periodized training approaches. In this model, blocks build upon each other in a predetermined order, and while individual block duration might flex slightly, the sequence remains constant.

Nonlinear progression allows blocks to rotate or alternate based on defined patterns. This approach supports training where multiple attributes need to be maintained simultaneously, with focus shifting between them according to specified rules.

In both progression types, blocks can be modified while a plan is active, but changes must maintain the integrity of the chosen progression strategy.

2. Training Block

Training blocks are the fundamental units of periodization within a plan. Each block represents a focused period of training with specific primary and maintenance objectives. Blocks bridge the gap between high-level training strategy and day-to-day workout implementation.

The key characteristic of a block is its focused nature - each block emphasizes specific training objectives while maintaining other attributes. This focus helps ensure systematic progress across different aspects of climbing performance over time.

Required Attributes:

- Unique identifier
- Primary training focus
- Duration (in weeks)

- Workouts (ordered list)

Block Management:
Blocks operate within specific constraints to maintain training effectiveness. Duration limits (1-6 weeks) ensure blocks are long enough to produce training effects but short enough to maintain focus and motivation. Within these constraints, block timing can be adjusted to accommodate external factors like conditions or recovery needs.

Workout sequencing within blocks is intentional - while individual workout dates can shift, the sequence of workouts should maintain training logic (e.g., appropriate spacing of high-intensity sessions).

3. Workout

Workouts represent single training sessions and serve as the bridge between planning and execution. Each workout consists of an ordered sequence of exercises designed to serve the block's objectives.

The workout level is where planning meets reality - while higher levels deal with organization and strategy, workouts handle the practical implementation of training. This includes managing actual versus planned timing, tracking completion, and recording training outcomes.

Required Attributes:

- Unique identifier
- Planned date
- Status: planned | completed | skipped
- Exercises (ordered list)

Workout Management:
Workouts must balance adherence to the training plan with flexibility for real-world constraints. While workouts can be rescheduled within their block, the system maintains the integrity of workout sequencing to preserve training logic.

Status tracking at the workout level provides crucial feedback for adjusting training plans. The distinction between planned and actual dates helps identify

patterns in schedule adherence that might suggest needed adjustments to the overall plan.

4. Exercise

Exercises are the atomic units of training - the specific activities that make up a workout. Each exercise has a defined type that determines its parameters and how it is tracked.

The exercise level handles the concrete details of training prescription and execution. This includes specific parameters like sets, reps, or duration, as well as tracking actual versus planned performance.

Required Attributes:

- Unique identifier
- Type identifier
- Base parameters (sets/reps/duration as appropriate for type)

Status Tracking:
Exercise tracking captures both completion status and actual versus planned parameters. This granular tracking provides the raw data needed for higher-level analysis of training effectiveness and progression.

Templates

The template system allows for efficient reuse of training structures at multiple levels. Templates capture the structure and relationships of training elements without specific dates or instance-specific details.

Templates can exist at the plan, block, or workout level. When a template is instantiated, all dates are set relative to a specified start date, and relationships between elements are maintained. This allows for efficient creation of new training instances while preserving proven structures.

Progress Tracking

Progress tracking occurs at multiple levels, with each level focusing on appropriate metrics for that scope. This hierarchical approach to tracking allows

for both detailed analysis and high-level overview of training progress.

Plan-level tracking focuses on overall adherence and progression through blocks. Block-level tracking emphasizes progress in primary focus areas and workout completion rates. Workout-level tracking handles the details of exercise completion and parameter achievement.

Core Requirements

The system maintains data integrity through careful management of relationships between levels. Changes at higher levels must maintain consistency at lower levels - for example, moving a block must preserve workout sequencing within that block.

All entities must maintain their parent-child relationships, and each level has minimum content requirements (e.g., plans must contain at least one block). These requirements ensure the system maintains meaningful training structures while remaining flexible enough to accommodate different training approaches.

```
erDiagram
    users ||--o{ training_plans : creates
    training_plans ||--|{ training_blocks : contains
    training_blocks ||--|{ workouts : contains
    workouts ||--|{ workout_exercises : contains
    exercise_types ||--o{ workout_exercises : defines
    workout_exercises ||--o{ exercise_logs : tracks

    users {
        uuid id PK
        string access_key "Random generated key"
        string nickname "Optional display name"
        timestamp last_access
        timestamp created_at
        timestamp updated_at
    }

    training_plans {
```

```
        uuid id PK
        uuid user_id FK
        string name
        string progression_type
        date start_date
        date end_date
        integer target_weekly_hours
        timestamp created_at
        timestamp updated_at
    }

    training_blocks {
        uuid id PK
        uuid plan_id FK
        string name
        string primary_focus
        integer duration_weeks
        integer sequence_order
        timestamp created_at
        timestamp updated_at
    }

    workouts {
        uuid id PK
        uuid block_id FK
        string name
        date planned_date
        date actual_date
        string status
        integer sequence_order
        timestamp created_at
        timestamp updated_at
    }

    exercise_types {
        uuid id PK
```

```
        string name
        string category
        jsonb parameters
        timestamp created_at
        timestamp updated_at
    }

    workout_exercises {
        uuid id PK
        uuid workout_id FK
        uuid exercise_type_id FK
        integer sequence_order
        jsonb planned_parameters
        timestamp created_at
        timestamp updated_at
    }

    exercise_logs {
        uuid id PK
        uuid workout_exercise_id FK
        jsonb actual_parameters
        text notes
        timestamp created_at
    }
```