`bookings` vs `dateAvailability` for courts

Even without enforcing overlaps, these two concepts serve *different layers* of the scheduling model:

`dateAvailability` = **capacity rule / constraint**

Think of `dateAvailability` as **a declarative override to the court's "normal hours"** for a specific date (or date range), usually used for:

- special opening/closing days (holiday hours)
- planned maintenance windows (court resurfacing, net replacement)
- administrative closures (venue event, staffing limitations)
- practice blocks that are *policy-driven* (e.g., "courts reserved for team practice 4–6pm")

In other words: `dateAvailability` answers *"is the court allowed to be scheduled then?"* and typically belongs to the *availability engine*.

`bookings` = **consumption of available capacity**

`bookings` are concrete allocations of time to a purpose or owner:

- a specific maintenance booking (work order)
- a reserved practice slot
- a blocked hold for an event (tournament desk hold, private rental)
- (potentially) match allocations (though tournaments often model match scheduling separately)

So: bookings answer *"someone is actually using/holding this time"* and belong to the *reservation ledger*.

How they relate (the most robust model)

A clean, deterministic precedence model is:

1. **Start with the court's normal daily window**: `[court.startTime, court.endTime)` for that date.
2. **Apply `dateAvailability`** to modify that window:
    - "unavailable" blocks subtract from the window
    - "available" blocks can either:

- *extend* the daily window (if you allow “open earlier / stay later”), or
- be treated as “allowed segments” (if you support whitelist-style availability)

3. **Subtract `bookings`** (since they consume time within the allowed window).

4. Remaining segments = **schedulable**.

This keeps `dateAvailability` as *policy* and `bookings` as *actual usage*. It also prevents a common anti-pattern where “availability” is encoded only as bookings, which makes operating hours, holidays, and global closures harder to reason about.

`dateAvailability` relative to `court.startTime` / `court.endTime`

Given the understanding (court has a daily availability window, within which `dateAvailability` and bookings exist), the most consistent interpretation is:

- `startTime` / `endTime` define the **default daily open interval** for that court.
- `dateAvailability` entries are **exceptions/overrides for specific dates** which are evaluated *inside* (or against) that daily interval.

The schema is intended to support:

`dateAvailability` = delta / override

- Default = `[startTime, endTime)`
- `dateAvailability` adds/removes segments for that date.
- This is the most natural fit for “normal hours + exceptions.”

Recommendation: expand Venue attributes (facility defaults + global closures)

Why this matters (sports facility reality)

Facilities are frequently closed or constrained independent of any particular court: staffing, building access, weather closures (for outdoor), events, seasonal hours, etc. A venue-level default prevents:

- duplicating hours across courts
- accidentally scheduling courts when the facility is closed
- inconsistent court configurations inside a single facility

Venue attribute set I recommend

Core defaults

- `openTime` (Daily opening time for the venue (HH:mm)
- `closeTime` (Daily closing time for the venue (HH:mm)
- `defaultStartTime`, `defaultEndTime` (daily open window for the facility)
- `timezone` (strongly recommended for correctness)
- `defaultBookingPolicies` (optional: min/max duration, buffer, lead time)
- `dateAvailability` (facility-level overrides: holidays, event closures, partial open days)
- `venueBookings` (optional: venue-wide holds that effectively block all courts, e.g., “power outage 9–12”)

Precedence

- Effective court availability = `venue defaults` ∩ `court hours` with both modified by their dateAvailability, then subtract bookings.

Design document: Venue-level defaults and constraints

1. Problem statement

The current model places daily operating hours and exception logic on **courts** but not on **venues**. This makes it possible for a court to be “available” even if the facility is closed, and forces duplication of hours and closures across courts.

2. Goals

- Represent facility operating hours and closures at the venue level.
- Allow courts to inherit defaults while still supporting court-specific overrides.
- Define deterministic interaction between:

    - venue defaults
    - court defaults
    - date-specific overrides
    - bookings

- Maintain backward compatibility (existing data should validate and behave as before).

3. Non-goals

- Redesign tournament match scheduling.
- Introduce multi-resource booking dependencies (e.g., staff + court + equipment), though schema should not prevent later extension.

4. Proposed data model

4.1 Venue default hours

Add `defaultStartTime` / `defaultEndTime` to Venue. Courts may omit `startTime` / `endTime` and inherit.

4.2 Venue date overrides

Add `dateAvailability` to Venue to represent:

- full-day closure
- partial closure (e.g., open 10:00–14:00 on a holiday)
- planned venue maintenance windows

4.3 Optional venue-wide bookings

Add `venueBookings` to Venue for “whole-facility” holds that should block all courts.

4.4 Timezone correctness

Add `timezone` to Venue (or Tournament). Venue is best because multiple venues can exist.

5. Scheduling semantics

For a given `venueId`, `courtId`, and `date`:

1. Determine base venue open window:
    - Use `venue.defaultStartTime/defaultEndTime` (if present)

2. Determine base court open window:
    - Use `court.startTime/endTime` if present, else inherit venue defaults

3. Establish base window:
    - `base = intersection(venueBase, courtBase)` (if one missing, treat as the other)

4. Apply dateAvailability:
    - Apply venue `dateAvailability` to `base` producing `base'`
    - Apply court `dateAvailability` to `base'` producing `allowed`

5. Subtract bookings:
    - Subtract venue bookings (for that date/resource scope) from `allowed`
    - Subtract court bookings from remaining

6. Result is the set of schedulable segments

Adjacency allowed: end==start is valid; overlap predicate uses half-open intervals `[start,end)`.

6. Backward compatibility

- Existing tournaments without venue defaults continue to behave as today.
- Courts with explicit `startTime/endTime` remain authoritative.
- Venue dateAvailability is additive; if absent, no change.

7. Validation and invariants

- `defaultEndTime` must be >= `defaultStartTime` (or strictly greater if you disallow zero hours).
- `dateAvailability` and `bookings/venueBookings` should not overlap *within the same object* if the no-overlap invariant is required (optional but recommended).
- When both venue and court hours exist, courts must not be “open” outside venue hours unless explicitly allowed via a policy flag (default should be “no”).