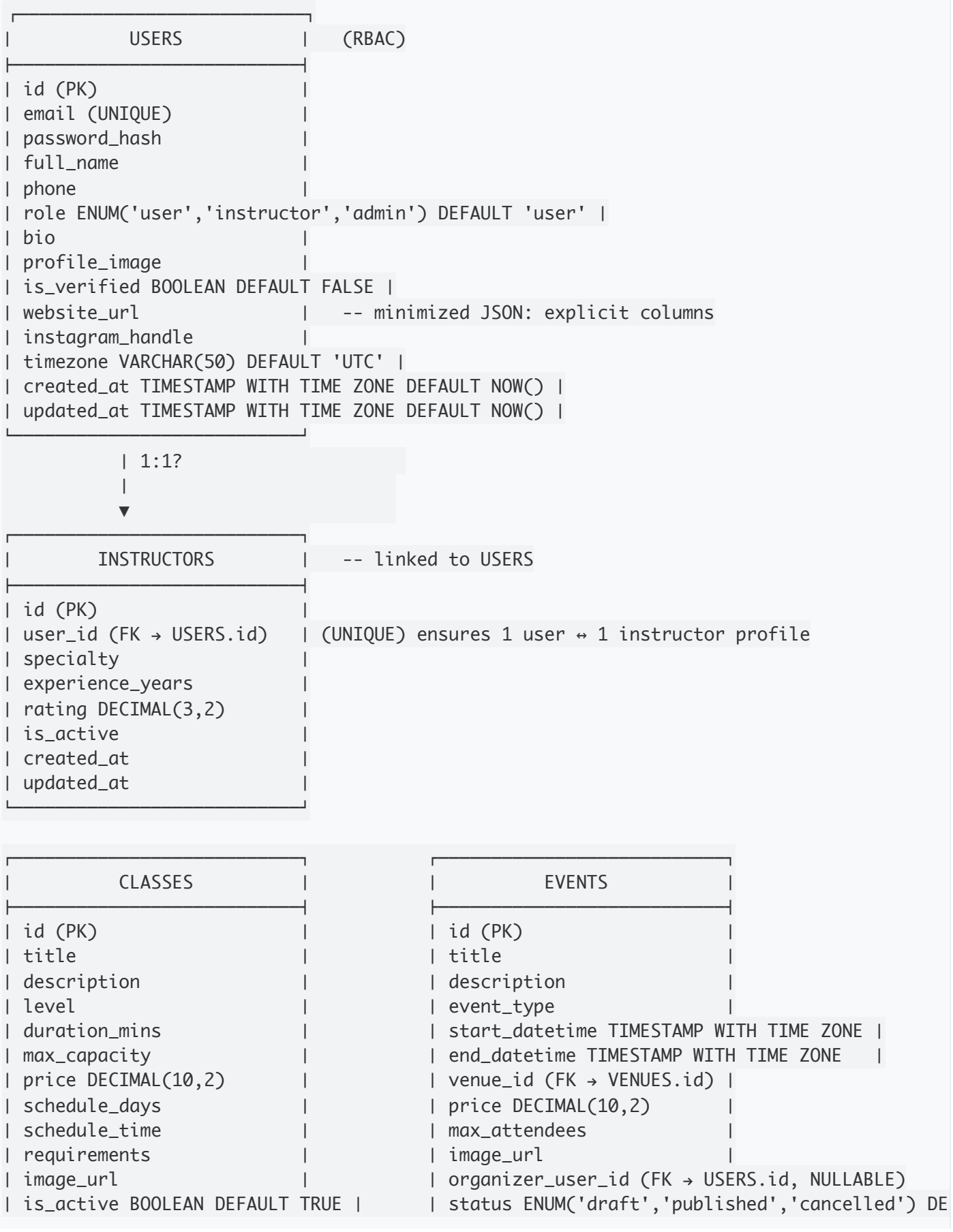
Dance Website - ERD v2 (RBAC, Venues, Styles, Unified Bookings, Transactions)

Overview

This ERD updates the prior schema to: - Merge ADMINS into USERS via RBAC (no ADMINS table) - Link INSTRUCTORS to USERS for unified accounts - Add VENUES and reference from EVENTS via venue_id - Normalize dance styles via mapping tables (EVENT_STYLES, CLASS_STYLES, USER_STYLES) - Unify BOOKINGS across classes/events with duplicate-prevention - Add TRANSACTIONS for payment/refund logging - Remove duplicated forum tables and add ON DELETE/UPDATE rules - Minimize JSON fields for MVP; prefer normalized columns - Add indexes/uniques for FKs, timestamps, start_time, and style mappings - **Add timezone support: store all timestamps in UTC with timezone fields** - **Add proper status/timestamp constraints and defaults** - **Implement soft delete via deleted_at fields for key tables** - **Optimize counters: aggregate from BOOKINGS instead of storing/updating counters**

Core Entities and Relationships



```
| deleted_at TIMESTAMP WITH TIME ZONE NULL |      | is_featured BOOLEAN DEFAULT FALSE |
| created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() | | deleted_at TIMESTAMP WITH TIME ZO
| updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
└──────────────────────────────┘           | created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW
           |                                | updated_at TIMESTAMP WITH TIME ZONE DEFAULT NO
           |                                └──────────────────────────────┘
           ▼

┌──────────────────────────────┐
|      CLASS_SESSIONS          |   -- for scheduling exceptions & specific sessions
├──────────────────────────────┤
| id (PK)                      |
| class_id (FK → CLASSES.id)|
| session_date DATE            |
| start_time TIME              |
| end_time TIME                |
| venue_id (FK → VENUES.id, NULLABLE) | -- override class default venue
| instructor_id (FK → INSTRUCTORS.id, NULLABLE) | -- override or substitute instructor
| max_capacity INTEGER         |           -- override class default capacity
| status ENUM('scheduled','cancelled','completed','rescheduled') DEFAULT 'scheduled' |
| cancellation_reason TEXT  |
| notes TEXT                   |           -- session-specific notes
| price_override DECIMAL(10,2) |       -- override class default price
| deleted_at TIMESTAMP WITH TIME ZONE NULL |       -- soft delete support
| created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
| updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
└──────────────────────────────┘
Constraints:
- UNIQUE (class_id, session_date, start_time)  -- prevent duplicate sessions
- CHECK (end_time > start_time)  -- ensure logical time ordering
- CHECK (max_capacity > 0)  -- ensure positive capacity
           |                                    |
           |                                    |
           ▼                                    ▼

┌──────────────────────────────┐          ┌──────────────────────────────┐
|    CLASS_INSTRUCTORS         |  (M:N)   |          VENUES              |
├──────────────────────────────┤          ├──────────────────────────────┤
| id (PK)                      |          | id (PK)                      |
| class_id (FK → CLASSES.id)|          | name                        |
| instructor_id (FK → INSTRUCTORS.id)     | address_line1                |
| is_primary                   |          | address_line2                |
| created_at                   |          | city                         |
| updated_at                   |          | state                        |
└──────────────────────────────┘          | country                      |
                                          | postal_code                  |
                                          | latitude DECIMAL(9,6)        |
                                          | longitude DECIMAL(9,6)       |
                                          | phone                        |
                                          | website_url                  |
                                          | created_at                   |
                                          | updated_at                   |
                                          └──────────────────────────────┘
```

```
┌──────────────────────────────┐
|         BOOKINGS             |   -- unified for class/event
├──────────────────────────────┤
| id (PK)                      |
| user_id (FK → USERS.id)      |
| class_id (FK → CLASSES.id, NULLABLE) |
| event_id (FK → EVENTS.id,  NULLABLE) |
| class_session_id (FK → CLASS_SESSIONS.id, NULLABLE) |
| booking_datetime TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
| status ENUM('pending','confirmed','cancelled','completed','refunded') DEFAULT 'pending'
| amount_paid DECIMAL(10,2) |
| payment_method               |
| notes                        |
| deleted_at TIMESTAMP WITH TIME ZONE NULL |
| created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
| updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
└──────────────────────────────┘
Constraints:
- CHECK (class_id IS NOT NULL OR event_id IS NOT NULL OR class_session_id IS NOT NULL)  -
- CHECK (((class_id IS NOT NULL)::int + (event_id IS NOT NULL)::int + (class_session_id I
- UNIQUE (user_id, class_id) WHERE class_id IS NOT NULL     -- prevent duplicates
- UNIQUE (user_id, event_id) WHERE event_id IS NOT NULL
- UNIQUE (user_id, class_session_id) WHERE class_session_id IS NOT NULL


┌──────────────────────────────┐
|       TRANSACTIONS           |   -- payment/refund logs
├──────────────────────────────┤
| id (PK)                      |
| booking_id (FK → BOOKINGS.id, NULLABLE) |
| user_id (FK → USERS.id)      |
| provider ENUM('stripe','paypal','other') |
| provider_payment_id          |
| provider_refund_id           | (NULLABLE)
| type ENUM('payment','refund','adjustment') |
| status ENUM('created','succeeded','failed','refunded','cancelled') |
| amount                       |
| currency                     |
| payload TEXT                 |  -- raw provider payload (minimize JSON usage)
| created_at                   |
| updated_at                   |
└──────────────────────────────┘


┌──────────────────────────────┐           ┌──────────────────────────────┐
|        DANCE_STYLES          |           |       USER_STYLES            |
├──────────────────────────────┤           ├──────────────────────────────┤
| id (PK)                      |           | id (PK)                      |
| name (UNIQUE)                |           | user_id (FK → USERS.id)      |
| category                     |           | style_id (FK → DANCE_STYLES.id) |
| is_active                    |           | proficiency ENUM('beginner','intermediate','adv
| created_at                   |           | created_at                   |
| updated_at                   |           | updated_at                   |
```

```
└──────────────────────────────┘          └──────────────────────────────┘
```

```
┌──────────────────────────────┐          ┌──────────────────────────────┐
│       CLASS_STYLES           │          │       EVENT_STYLES           │
├──────────────────────────────┤          ├──────────────────────────────┤
│ id (PK)                      │          │ id (PK)                      │
│ class_id (FK → CLASSES.id)   │          │ event_id (FK → EVENTS.id)    │
│ style_id (FK → DANCE_STYLES.id)│        │ style_id (FK → DANCE_STYLES.id)│
│ created_at                   │          │ created_at                   │
│ updated_at                   │          │ updated_at                   │
└──────────────────────────────┘          └──────────────────────────────┘
```

Constraints:
- UNIQUE (class_id, style_id)
- UNIQUE (event_id, style_id)
- UNIQUE (user_id, style_id) in USER_STYLES

```
┌──────────────────────────────┐          ┌──────────────────────────────┐
│        FORUM_POSTS           │          │        FORUM_REPLIES         │
├──────────────────────────────┤          ├──────────────────────────────┤
│ id (PK)                      │          │ id (PK)                      │
│ user_id (FK → USERS.id)      │          │ post_id (FK → FORUM_POSTS.id)
│ category                     │          │ user_id (FK → USERS.id)      │
│ title                        │          │ parent_id (FK → FORUM_REPLIES.id, NULLABLE)
│ content                      │          │ content                      │
│ views_count                  │          │ likes_count                  │
│ likes_count                  │          │ is_solution                  │
│ replies_count                │          │ created_at                   │
│ is_pinned                    │          │ updated_at                   │
│ is_locked                    │          └──────────────────────────────┘
│ created_at                   │
│ updated_at                   │
└──────────────────────────────┘
```

```
┌──────────────────────────────┐          ┌──────────────────────────────┐
│       NOTIFICATIONS          │          │         AUDIT_LOGS           │
├──────────────────────────────┤          ├──────────────────────────────┤
│ id (PK)                      │          │ id (PK)                      │
│ user_id (FK → USERS.id)      │          │ user_id (FK → USERS.id)      │  -- replaces admin_
│ type                         │          │ action                       │
│ title                        │          │ table_name                   │
│ message                      │          │ record_id                    │
│ is_read                      │          │ old_values TEXT              │
│ priority                     │          │ new_values TEXT              │
│ action_url                   │          │ ip_address                   │
│ created_at                   │          │ user_agent                   │
│ updated_at                   │          │ created_at                   │
└──────────────────────────────┘          │ updated_at                   │
                                          └──────────────────────────────┘
```

```
┌──────────────────────────────┐          ┌──────────────────────────────┐
│      CONTACT_MESSAGES        │          │        TESTIMONIALS          │
├──────────────────────────────┤          ├──────────────────────────────┤
```

```
| id (PK)                    |           | id (PK)                    |
| name                       |           | user_id (FK → USERS.id)    |
| email                      |           | rating                     |
| phone                      |           | message                    |
| subject                    |           | is_featured                |
| message                    |           | created_at                 |
| is_read                    |           | updated_at                 |
| admin_response             |
| created_at                 |
| updated_at                 |
└────────────────────────────┘

┌────────────────────────────┐           ┌────────────────────────────┐
|      PARTNER_REQUESTS      |           |       PARTNER_MATCHES      |
├────────────────────────────┤           ├────────────────────────────┤
| id (PK)                    |           | id (PK)                    |
| requester_id (FK → USERS.id) |         | user1_id (FK → USERS.id)   |
| skill_level                |           | user2_id (FK → USERS.id)   |
| location_city              |           | match_score                |
| availability_text          |           | status                     |
| message                    |           | created_at                 |
| status                     |           | updated_at                 |
| created_at                 |           └────────────────────────────┘
| updated_at                 |
└────────────────────────────┘

┌────────────────────────────┐
|        FAVORITES           |   -- polymorphic favorites system
├────────────────────────────┤
| id (PK)                    |
| user_id (FK → USERS.id)    |
| entity_type ENUM('event','class','instructor','venue','user') |
| entity_id BIGINT           |  -- references id in various tables
| created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
| updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW() |
└────────────────────────────┘
Constraints:
- UNIQUE (user_id, entity_type, entity_id)  -- prevent duplicate favorites
- CHECK (entity_id > 0)  -- ensure valid entity reference
```

Key Changes vs Previous ERD

1. Removed ADMINS table. USERS has role and all admin actions reference USERS.
2. INSTRUCTORS has user_id (unique), removing duplicate identity fields from instructors; social links are simplified to explicit columns on USERS.
3. Added VENUES and referenced by EVENTS via venue_id. Removed free-text location/address from EVENTS (moved to VENUES).

4. Normalized dance styles with mapping tables: USER_STYLES, CLASS_STYLES, EVENT_STYLES.
5. Unified BOOKINGS with constraints to prevent duplicate bookings per user per class/event.
6. Added TRANSACTIONS to log payments/refunds with provider IDs and status.
7. Forum tables are singular (FORUM_POSTS, FORUM_REPLIES). All FKs include ON DELETE/UPDATE rules below.
8. Minimized JSON fields: replaced preferences/tags/social_links with normalized columns or TEXT where appropriate.
9. Added indexes and unique constraints for performance and data integrity.

Foreign Keys and Referential Actions

Unless otherwise noted, FKs use: - ON DELETE CASCADE where child records have no meaning without parent (e.g., CLASS_INSTRUCTORS, STYLES mappings, BOOKINGS, TRANSACTIONS linked to BOOKINGS) - ON UPDATE CASCADE for id changes (rare but safe) - ON DELETE SET NULL where history should be retained without a parent (e.g., TRANSACTIONS.booking_id, EVENTS.organizer_user_id)

Examples: - INSTRUCTORS.user_id → USERS.id ON DELETE CASCADE ON UPDATE CASCADE - CLASS_INSTRUCTORS.class_id → CLASSES.id ON DELETE CASCADE ON UPDATE CASCADE - CLASS_INSTRUCTORS.instructor_id → INSTRUCTORS.id ON DELETE CASCADE ON UPDATE CASCADE - EVENTS.venue_id → VENUES.id ON DELETE RESTRICT ON UPDATE CASCADE - BOOKINGS.user_id → USERS.id ON DELETE CASCADE ON UPDATE CASCADE - BOOKINGS.class_id → CLASSES.id ON DELETE CASCADE ON UPDATE CASCADE - BOOKINGS.event_id → EVENTS.id ON DELETE CASCADE ON UPDATE CASCADE - TRANSACTIONS.booking_id → BOOKINGS.id ON DELETE SET NULL ON UPDATE CASCADE - TRANSACTIONS.user_id → USERS.id ON DELETE CASCADE ON UPDATE CASCADE - BOOKINGS.class_session_id → CLASS_SESSIONS.id ON DELETE CASCADE ON UPDATE CASCADE - USER_STYLES.user_id → USERS.id ON DELETE CASCADE; style_id → DANCE_STYLES.id ON DELETE RESTRICT - CLASS_STYLES.class_id → CLASSES.id ON DELETE CASCADE; style_id → DANCE_STYLES.id ON DELETE RESTRICT - EVENT_STYLES.event_id → EVENTS.id ON DELETE CASCADE; style_id → DANCE_STYLES.id ON DELETE RESTRICT - FORUM_POSTS.user_id → USERS.id ON DELETE SET NULL (optional, if you want to retain posts) or CASCADE if you prefer deletion - FORUM_REPLIES.user_id → USERS.id ON DELETE SET NULL; post_id → FORUM_POSTS.id ON DELETE CASCADE; parent_id → FORUM_REPLIES.id ON DELETE CASCADE - FAVORITES.user_id → USERS.id ON DELETE CASCADE ON UPDATE CASCADE - CLASS_SESSIONS.class_id → CLASSES.id ON DELETE CASCADE ON UPDATE CASCADE -

CLASS_SESSIONS.venue_id → VENUES.id ON DELETE SET NULL ON UPDATE CASCADE - CLASS_SESSIONS.instructor_id → INSTRUCTORS.id ON DELETE SET NULL ON UPDATE CASCADE

Indexes and Constraints

Single Column Indexes

- USERS(email) UNIQUE
- USERS(role), USERS(created_at) INDEX
- INSTRUCTORS(user_id) UNIQUE, INDEX
- CLASSES(created_at), CLASSES(is_active) INDEX
- EVENTS(venue_id) INDEX, EVENTS(status) INDEX
- VENUES(name), VENUES(city), VENUES(country) INDEX
- BOOKINGS(user_id) INDEX, BOOKINGS(created_at) INDEX, BOOKINGS(status) INDEX
- BOOKINGS(class_session_id) INDEX
- TRANSACTIONS(provider, provider_payment_id) UNIQUE (nullable provider_refund_id can be separate unique when present)
- FORUM_POSTS(created_at), FORUM_REPLIES(created_at) INDEX

Composite Indexes for Query Performance

Event Location & Time Queries: - `CREATE INDEX idx_events_venue_datetime ON EVENTS (venue_id, start_datetime)` - `CREATE INDEX idx_events_venue_status_datetime ON EVENTS (venue_id, status, start_datetime)` - `CREATE INDEX idx_venues_city_country ON VENUES (city, country)`

City-based Event Searches (via VENUES join): - `CREATE INDEX idx_venues_city_datetime ON VENUES (city, created_at)` -- for venue discovery - Combined with EVENTS indexes above for efficient city + date queries

User Activity & Booking Patterns: - `CREATE INDEX idx_bookings_user_datetime ON BOOKINGS (user_id, booking_datetime)` - `CREATE INDEX idx_bookings_status_datetime ON BOOKINGS (status, created_at)` - `CREATE INDEX idx_transactions_user_datetime ON TRANSACTIONS (user_id, created_at)` - `CREATE INDEX idx_transactions_status_datetime ON TRANSACTIONS (status, created_at)`

Style-based Searches: - `CREATE INDEX idx_user_styles_style_proficiency ON USER_STYLES (style_id, proficiency)` - `CREATE INDEX idx_class_styles_style_id ON CLASS_STYLES (style_id, class_id)` - `CREATE INDEX idx_event_styles_style_id ON EVENT_STYLES (style_id, event_id)`

Forum & Content Queries: - `CREATE INDEX idx_forum_posts_category_datetime ON FORUM_POSTS (category, created_at)` - `CREATE INDEX idx_forum_posts_user_datetime ON FORUM_POSTS (user_id, created_at)` - `CREATE INDEX idx_forum_replies_post_datetime ON FORUM_REPLIES (post_id, created_at)`

Favorites & User Preferences: - `CREATE INDEX idx_favorites_user_entity ON FAVORITES (user_id, entity_type)` - `CREATE INDEX idx_favorites_entity_type_id ON FAVORITES (entity_type, entity_id)` - `CREATE INDEX idx_favorites_user_created ON FAVORITES (user_id, created_at)`

Class Sessions & Scheduling: - `CREATE INDEX idx_class_sessions_class_date ON CLASS_SESSIONS (class_id, session_date)` - `CREATE INDEX idx_class_sessions_date_status ON CLASS_SESSIONS (session_date, status)` - `CREATE INDEX idx_class_sessions_venue_date ON CLASS_SESSIONS (venue_id, session_date)` - `CREATE INDEX idx_class_sessions_instructor_date ON CLASS_SESSIONS (instructor_id, session_date)`

Unique Constraints

- USER_STYLES(user_id, style_id)
- CLASS_STYLES(class_id, style_id)
- EVENT_STYLES(event_id, style_id)
- BOOKINGS(user_id, class_id) where class_id not null
- BOOKINGS(user_id, event_id) where event_id not null
- BOOKINGS(user_id, class_session_id) where class_session_id not null
- FAVORITES(user_id, entity_type, entity_id)
- CLASS_SESSIONS(class_id, session_date, start_time)

Query Optimization Examples

Efficient City + Date Queries

The composite indexes enable fast execution of common search patterns:

Example 1: Find events in a specific city on a date range

```
-- Optimized by: idx_events_venue_datetime + idx_venues_city_country
SELECT e.*, v.name as venue_name, v.city
FROM EVENTS e
JOIN VENUES v ON e.venue_id = v.id
WHERE v.city = 'New York'
  AND e.start_datetime BETWEEN '2024-03-01' AND '2024-03-31'
  AND e.status = 'published';
```

Example 2: Find available events at a specific venue

```
-- Optimized by: idx_events_venue_status_datetime
SELECT * FROM EVENTS
WHERE venue_id = 123
  AND status = 'published'
  AND start_datetime > NOW()
ORDER BY start_datetime;
```

Example 3: User booking history with recent first

```
-- Optimized by: idx_bookings_user_datetime
SELECT * FROM BOOKINGS
WHERE user_id = 456
ORDER BY booking_datetime DESC
LIMIT 20;
```

Example 4: Find users by dance style and skill level

```
-- Optimized by: idx_user_styles_style_proficiency
SELECT u.*, us.proficiency
FROM USERS u
JOIN USER_STYLES us ON u.id = us.user_id
WHERE us.style_id = 789
  AND us.proficiency = 'intermediate';
```

Example 5: Event discovery by style

```
-- Optimized by: idx_event_styles_style_id + idx_events_venue_datetime
SELECT e.*, v.city
FROM EVENTS e
JOIN EVENT_STYLES es ON e.id = es.event_id
JOIN VENUES v ON e.venue_id = v.id
WHERE es.style_id = 789
  AND e.start_datetime > NOW()
  AND e.status = 'published'
ORDER BY e.start_datetime;
```

Example 6: User's favorite events with venue details

```
-- Optimized by: idx_favorites_user_entity + idx_events_venue_datetime
SELECT e.*, v.name as venue_name, v.city, f.created_at as favorited_at
FROM FAVORITES f
JOIN EVENTS e ON f.entity_id = e.id
JOIN VENUES v ON e.venue_id = v.id
WHERE f.user_id = 456
  AND f.entity_type = 'event'
  AND e.status = 'published'
ORDER BY f.created_at DESC;
```

Example 7: Find upcoming class sessions with venue and instructor details

```
-- Optimized by: idx_class_sessions_date_status
SELECT
    c.title as class_title,
    cs.session_date,
    cs.start_time,
    cs.end_time,
    cs.status,
    COALESCE(v_session.name, v_class.name) as venue_name,
    COALESCE(i_session.name, i_default.name) as instructor_name,
    cs.notes,
    COALESCE(cs.price_override, c.price) as session_price
FROM CLASS_SESSIONS cs
JOIN CLASSES c ON cs.class_id = c.id
LEFT JOIN VENUES v_session ON cs.venue_id = v_session.id  -- session-specific venue
LEFT JOIN VENUES v_class ON c.venue_id = v_class.id       -- class default venue
LEFT JOIN INSTRUCTORS i_session ON cs.instructor_id = i_session.id  -- session instructor
LEFT JOIN CLASS_INSTRUCTORS ci ON c.id = ci.class_id AND ci.is_primary = true
LEFT JOIN INSTRUCTORS i_default ON ci.instructor_id = i_default.id   -- default instructo
WHERE cs.session_date >= CURRENT_DATE
  AND cs.status IN ('scheduled', 'rescheduled')
ORDER BY cs.session_date, cs.start_time;
```

Index Usage Patterns

- **Leading column optimization**: Indexes work best when queries filter on the leftmost columns first
- **Range queries**: Composite indexes with datetime as the rightmost column optimize date range filters
- **Covering indexes**: Some indexes may include additional columns to avoid table lookups
- **Maintenance**: Monitor query performance and adjust indexes based on actual usage patterns

Data Types and Constraints with Timezone Support

Timezone Handling Strategy

- **All timestamps stored in UTC** using `TIMESTAMP WITH TIME ZONE`
- **User timezone preferences** stored in USERS.timezone field
- **Event datetime fields** use timezone-aware timestamps for accurate scheduling
- **Application layer** handles timezone conversion for display

Common Field Types with Constraints

- **id**: `BIGINT AUTO_INCREMENT PRIMARY KEY`
- **email**: `VARCHAR(255) UNIQUE NOT NULL`
- **password_hash**: `VARCHAR(255) NOT NULL`
- **created_at/updated_at**: `TIMESTAMP WITH TIME ZONE DEFAULT NOW()` with auto-update triggers
- **boolean flags**: `BOOLEAN DEFAULT FALSE/TRUE` with explicit defaults
- **status enums**: Proper ENUM types with DEFAULT values and CHECK constraints
- **price/amount fields**: `DECIMAL(10,2)` for precise monetary calculations
- **rating**: `DECIMAL(3,2) DEFAULT 0.00 CHECK (rating >= 0.00 AND rating <= 5.00)`
- **timezone**: `VARCHAR(50) DEFAULT 'UTC'` with CHECK constraint for valid timezone names

Timestamp Constraints and Defaults

- **USERS**:
- `created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()`

- `updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()` (auto-update on change)
- **EVENTS**:
- `start_datetime TIMESTAMP WITH TIME ZONE NOT NULL`
- `end_datetime TIMESTAMP WITH TIME ZONE NOT NULL`
- `CHECK (end_datetime > start_datetime)` -- ensure logical datetime ordering
- **BOOKINGS**:
- `booking_datetime TIMESTAMP WITH TIME ZONE DEFAULT NOW()`
- `status DEFAULT 'pending'`
- **TRANSACTIONS**:
- `created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()`
- `status DEFAULT 'created'`

Status Field Constraints

- **USERS.role**: `ENUM('user','instructor','admin') DEFAULT 'user' NOT NULL`
- **EVENTS.status**: `ENUM('draft','published','cancelled') DEFAULT 'draft' NOT NULL`
- **BOOKINGS.status**: `ENUM('pending','confirmed','cancelled','completed','refunded') DEFAULT 'pending' NOT NULL`
- **TRANSACTIONS.status**: `ENUM('created','succeeded','failed','refunded','cancelled') DEFAULT 'created' NOT NULL`
- **TRANSACTIONS.type**: `ENUM('payment','refund','adjustment') NOT NULL`

Database Triggers for Timestamp Management

- **Auto-update triggers** for `updated_at` fields on all tables
- **Timezone validation triggers** to ensure USERS.timezone contains valid timezone identifiers
- **Status transition triggers** to enforce valid state changes (e.g., can't go from 'completed' to 'pending')

Timezone Validation Examples

```
-- PostgreSQL example for timezone validation
ALTER TABLE USERS ADD CONSTRAINT valid_timezone
CHECK (timezone IN (SELECT name FROM pg_timezone_names));

-- MySQL example (simplified)
ALTER TABLE USERS ADD CONSTRAINT valid_timezone
CHECK (timezone REGEXP '^[A-Za-z_/]+$');
```

Event Datetime Constraints

- `CHECK (EVENTS.end_datetime > EVENTS.start_datetime)`
- `CHECK (EVENTS.start_datetime > NOW() - INTERVAL '1 day')` -- prevent scheduling far in past
- `CHECK (EVENTS.max_attendees > 0)`

Soft Delete Implementation

- **deleted_at**: `TIMESTAMP WITH TIME ZONE NULL` (present in CLASSES, EVENTS, CLASS_SESSIONS, BOOKINGS)
- Tables with `deleted_at` should have an index: `CREATE INDEX idx_{table}_deleted_at ON {TABLE} (deleted_at)`
- Query patterns should include `WHERE deleted_at IS NULL` to exclude soft-deleted records
- Views can be created to simplify queries: `CREATE VIEW active_{table} AS SELECT * FROM {TABLE} WHERE deleted_at IS NULL`

Counter Implementation

- Avoid storing and updating counters like `current_attendees` directly in tables
- Instead, calculate counts on-demand using subqueries or aggregation:

```
-- Example: Get event with attendee count
SELECT e.*, COUNT(b.id) as current_attendees
FROM EVENTS e
LEFT JOIN BOOKINGS b ON e.id = b.event_id AND b.status IN ('confirmed', 'pending') AND b.
WHERE e.id = 123 AND e.deleted_at IS NULL
GROUP BY e.id;

-- Example: Get class sessions with attendee counts
SELECT cs.*, COUNT(b.id) as current_attendees
FROM CLASS_SESSIONS cs
LEFT JOIN BOOKINGS b ON cs.id = b.class_session_id AND b.status IN ('confirmed', 'pending
WHERE cs.class_id = 456 AND cs.deleted_at IS NULL
GROUP BY cs.id;
```

- Create materialized views or refresh counts periodically for reports that need frequent access to counts
- Validate capacity constraints at application level: `COUNT(bookings) <= max_attendees`

Notes on JSON Minimization

- Replaced JSON social_links with explicit columns on USERS (website_url, instagram_handle). Add more as needed.
- Replaced EVENT.tags JSON with normalized styles mapping; additional tagging can be added later as TAGS/TAG_MAPPINGS if needed.
- AUDIT_LOGS uses TEXT for old/new values to avoid complex JSON; can move to JSON later as needed.

Suggested SQL Migration Outline (RDBMS-agnostic, adjust types)

- Add USERS.role, website_url, instagram_handle
- Drop ADMINS; migrate admin rows into USERS with role='admin'
- INSTRUCTORS: add user_id UNIQUE NOT NULL; backfill from existing email/user mapping; drop duplicate identity fields if present
- Create VENUES; add EVENTS.venue_id; backfill from existing location/address; drop EVENTS.location/address

- Create DANCE_STYLES, USER_STYLES, CLASS_STYLES, EVENT_STYLES; backfill from prior dance_style columns then drop those columns
- Create unified BOOKINGS; migrate CLASS_BOOKINGS and EVENT_BOOKINGS into BOOKINGS; drop old tables
- Create TRANSACTIONS; link to BOOKINGS where applicable
- Add deleted_at to CLASSES, EVENTS, CLASS_SESSIONS, BOOKINGS (all NULL initially)
- Remove current_attendees from EVENTS and CLASS_SESSIONS (use aggregation from BOOKINGS instead)
- Create indexes on deleted_at fields and updated indexes per above
- Add class_session_id to BOOKINGS and create associated constraint/index
- Ensure ON DELETE/UPDATE rules per above

Open Decisions

- FORUM_POSTS/REPLIES ON DELETE behavior: choose SET NULL vs CASCADE to align with content retention policy.
- EVENTS.organizer_user_id optional; you may also want ORGANIZERS table linked to USERS similar to INSTRUCTORS.
- If you need additional user preferences later, consider a USER_PREFERENCES table rather than JSON.
- Consider whether soft delete (deleted_at) should be added to additional tables beyond the core entities.
- Decide if any additional tables need materialized aggregation views for performance-critical counter access.