Nexus Family Pass - PostgreSQL Database Schema

Version: 1.0

Last Updated: January 2026

Database: PostgreSQL 15+

Author: Senior Database Architect

---

1. Conceptual Entity-Relationship Diagram

Core Entities and Relationships

```
┌──────────────────────────────────────────────────────────────────────
│                    NEXUS FAMILY PASS - ER DIAGRAM                    │
└──────────────────────────────────────────────────────────────────────

┌─────────────────┐    1:N    ┌─────────────────┐    N:1
┌─────────────────┐
│  CORPORATES   │◄──────────────│   USERS   │──────────────►│
VENUES    │
│  (Companies)   │          │ (Parents/HR/etc) │          │ (Activity Prov) │
└─────────────────┘          └─────────────────┘
└─────────────────┘
    │                  │                  │
    │ 1:N              │ 1:N              │ 1:N
    ▼                  ▼                  ▼
┌─────────────────┐          ┌─────────────────┐
┌─────────────────┐
│ SUBSCRIPTIONS  │         │  CHILDREN   │         │ ACTIVITIES  │
│ (Plans/Credits) │         │ (Child Profiles) │         │ (Activity Types) │
└─────────────────┘          └─────────────────┘
└─────────────────┘
    │                  │                  │
    │ 1:N              │ N:M              │ 1:N
    ▼                  ▼                  ▼
┌─────────────────┐          ┌─────────────────┐
┌─────────────────┐
│ CREDIT_LEDGER  │         │  BOOKINGS   │◄──────────────►│ ACTIVITY_SESSIONS│
│ (Transactions) │         │ (Reservations) │         │ (Time Slots)  │
└─────────────────┘          └─────────────────┘
└─────────────────┘
                  │
        ┌───────────────────────┼───────────────────────┐
        │         │         │
        ▼         ▼         ▼
    ┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
    │    WAITLIST     │   │    FEEDBACK     │   │     REVIEWS     │
    │ (Queue Entries) │   │ (Quick Rating)  │   │(Detailed Review)│
    └─────────────────┘   └─────────────────┘   └─────────────────┘


Additional Supporting Entities:
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
┌─────────────────┐
│  NOTIFICATIONS  │   │ VENUE_SCORES    │   │ AI_SUGGESTIONS  │   │   AUDIT_LOGS    │
│ (Multi-Channel) │   │(Weekly Computed)│   │(Monthly Curation)│   │  (Compliance)   │
└─────────────────┘   └─────────────────┘   └─────────────────┘
└─────────────────┘
```

Relationship Summary

Relationship	Type	Description
Corporate → Users	One-to-Many	Each company has multiple employee users
Corporate → Subscriptions	One-to-Many	Companies can have multiple subscription periods
User (Parent) → Children	One-to-Many	Parents can register multiple children
Venue → Activities	One-to-Many	Venues offer multiple activity types
Activity → Sessions	One-to-Many	Activities have multiple scheduled sessions
Session → Bookings	One-to-Many	Sessions can have multiple child bookings
Child → Bookings	One-to-Many	Children can have multiple bookings
Booking → Feedback	One-to-One	Each booking can have one quick feedback
Booking → Review	One-to-One	Each booking can have one detailed review
User → Notifications	One-to-Many	Users receive multiple notifications
Venue → Venue Scores	One-to-Many	Weekly performance score history
Child → Waitlist	One-to-Many	Children can be on multiple waitlists
Child → AI Suggestions	One-to-Many	Monthly curated suggestions per child

---

1.5 Pinecone Integration Strategy

Division of Responsibilities

Data Type	PostgreSQL	Pinecone	Rationale
Activity metadata	✅ Store	❌	Relational queries, joins with venues
Activity embeddings	Reference ID only	✅ Store vectors	Semantic similarity search
Child preferences	Reference ID only	✅ Store vectors	Personalization matching
NL search queries	Audit log only	✅ Query vectors	Intent-based search
Sibling compatibility	Reference ID only	✅ Multi-vector ops	Group matching algorithms
Booking transactions	✅ Full storage	❌	ACID compliance required
Reviews/Ratings	✅ Full storage	❌	Aggregations, reporting
Venue scores	✅ Full storage	Metadata filter	Filtering during search

Pinecone Index Structure (Recommended)

```
Index: nexus-activities
├── Namespace: activity-embeddings
│   └── Vectors: Activity description embeddings
│       └── Metadata: {activity_id, venue_id, category, min_age, max_age,
│                credits, is_messy, is_competitive, is_outdoor, venue_score}
│
├── Namespace: child-preferences
│   └── Vectors: Child interest/preference embeddings
│       └── Metadata: {child_id, parent_id, energy_level, social_pref, age}
│
└── Namespace: search-queries
    └── Vectors: Historical NL query embeddings (for improving search)
        └── Metadata: {query_id, user_id, resulted_in_booking, timestamp}
```

Sync Strategy

PostgreSQL remains the **source of truth** for all data. Pinecone vectors are derived and can be regenerated. The schema includes sync tracking columns to ensure consistency.

---

2. Detailed SQL Schema (DDL)

2.1 Extension Dependencies

```sql
-- ==========================================================================
-- NEXUS FAMILY PASS - DATABASE SCHEMA
-- PostgreSQL 15+ Required
-- ==========================================================================

-- Enable required PostgreSQL extensions for UUID generation and full-text search
CREATE EXTENSION IF NOT EXISTS "pgcrypto";     -- Provides gen_random_uuid() function for generating UUIDs
CREATE EXTENSION IF NOT EXISTS "pg_trgm";      -- Trigram extension for fuzzy text search capabilities
CREATE EXTENSION IF NOT EXISTS "btree_gin";    -- GIN index support for efficient JSONB querying
```

2.2 Enum Types (Domain-Specific Value Constraints)

```sql
```

```sql
-- ================================================================================
-- ENUM TYPE DEFINITIONS
-- Using PostgreSQL ENUM types ensures data integrity at the database level
-- and provides better query performance than VARCHAR with CHECK constraints
-- ================================================================================

-- User role enumeration defining the four primary user types in the system
-- Parent: End-user who books activities for their children
-- HR Admin: Corporate administrator managing employee benefits enrollment
-- Venue Admin: Activity provider managing their venue and listings
-- Platform Admin: System administrator with full platform access
CREATE TYPE user_role AS ENUM (
    'parent',           -- End-user parent who books activities for children
    'hr_admin',         -- HR administrator for corporate accounts
    'venue_admin',      -- Venue owner/manager who creates activities
    'platform_admin'    -- Nexus platform super administrator
);

-- Booking status tracking the complete lifecycle of a reservation
CREATE TYPE booking_status AS ENUM (
    'pending',          -- Booking created but awaiting venue confirmation
    'confirmed',        -- Booking confirmed by venue/system
    'completed',        -- Activity attended and marked complete
    'cancelled_parent', -- Cancelled by parent (refund rules apply)
    'cancelled_venue',  -- Cancelled by venue (full refund guaranteed)
    'no_show'           -- Child did not attend (credits forfeited)
);

-- Venue approval status for the venue onboarding workflow
CREATE TYPE venue_status AS ENUM (
    'pending_approval', -- Initial application submitted, awaiting review
    'active',           -- Approved and visible to parents
    'suspended',        -- Temporarily disabled due to issues
    'rejected'          -- Application denied (with feedback)
);

-- Credit transaction types for the financial ledger
CREATE TYPE credit_transaction_type AS ENUM (
    'allocation',       -- Monthly credits allocated from subscription
    'booking_debit',    -- Credits spent on booking an activity
    'refund_credit',    -- Credits returned due to cancellation (≥48hrs)
    'partial_refund',   -- Partial credits returned when venue modifies activity (PRD 2.2)
    'forfeit',          -- Credits forfeited (cancellation <48hrs or no-show)
    'adjustment',       -- Manual adjustment by platform admin
    'expiry'            -- Monthly unused credits expired
);
```

```
-- Notification delivery channels supported by the platform
CREATE TYPE notification_channel AS ENUM (
    'email',          -- Primary email notification
    'sms',            -- SMS text message
    'whatsapp',       -- WhatsApp business messaging
    'push',           -- Mobile app push notification
    'in_app'          -- In-application notification center
);

-- Notification priority levels for delivery and display ordering
CREATE TYPE notification_priority AS ENUM (
    'low',            -- Informational, can be batched in digests
    'normal',         -- Standard delivery timing
    'high',           -- Time-sensitive, deliver immediately
    'urgent'          -- Critical alerts requiring immediate action
);

-- Child energy level preference from onboarding quiz (Question 1)
CREATE TYPE energy_level AS ENUM (
    'calm_focused',   -- Prefers quiet, concentrated activities
    'balanced',       -- Moderate energy, flexible
    'high_energy'     -- Active, physical activities preferred
);

-- Child social preference from onboarding quiz (Question 2)
CREATE TYPE social_preference AS ENUM (
    'solo',           -- Prefers individual activities
    'small_group',    -- Comfortable in groups of 3-6
    'big_social'      -- Thrives in larger group settings
);

-- Activity category classification for filtering and recommendations
CREATE TYPE activity_category AS ENUM (
    'stem',           -- Science, Technology, Engineering, Math
    'arts_crafts',    -- Visual arts, crafting, creative projects
    'music',          -- Musical instruments, singing, composition
    'sports',         -- Physical sports and athletic activities
    'nature',         -- Outdoor exploration, environmental education
    'cooking',        -- Culinary skills, baking, food preparation
    'reading',        -- Literature, storytelling, book clubs
    'building',       -- Construction, LEGO, architecture
    'dance',          -- Dance styles, movement, choreography
    'drama'           -- Theater, acting, performance arts
);

-- Subscription plan tiers for corporate accounts
```

```sql
CREATE TYPE subscription_tier AS ENUM (
    'basic',            -- Entry-level plan with limited credits
    'standard',         -- Mid-tier with moderate credits
    'premium',          -- High-tier with premium venue access
    'enterprise'        -- Custom enterprise agreements
);

-- Cancellation reason categories (from UI Spec line 330)
CREATE TYPE cancellation_reason_type AS ENUM (
    'schedule_conflict',    -- Parent has scheduling conflict
    'child_sick',           -- Child is unwell
    'found_better_option',  -- Found a preferable activity
    'venue_cancelled',      -- Venue initiated cancellation
    'weather',              -- Weather-related cancellation
    'transportation',       -- Transportation issues
    'other'                 -- Other reason (with notes)
);

-- Time of day for session filtering (from UI Spec line 196)
CREATE TYPE time_of_day AS ENUM (
    'morning',          -- Before 12:00 PM
    'afternoon',        -- 12:00 PM - 5:00 PM
    'evening'           -- After 5:00 PM
);

-- Employee invitation status tracking
CREATE TYPE invitation_status AS ENUM (
    'pending',          -- Invitation sent, awaiting response
    'accepted',         -- Employee accepted and registered
    'expired',          -- Invitation link expired
    'revoked'           -- HR revoked the invitation
);
```

2.3 Core User and Authentication Tables

```
sql
```

```sql
-- ==========================================================================
-- CORPORATE ACCOUNTS TABLE
-- Stores B2B company information for the corporate subscription model
-- Each company purchases activity credits for their employees
-- ==========================================================================
CREATE TABLE corporates (
    -- Primary identifier using UUID for security and distributed systems compatibility
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Company legal name as registered for invoicing purposes
    company_name VARCHAR(255) NOT NULL,

    -- Trading name or brand name if different from legal name
    display_name VARCHAR(255),

    -- Corporate tax identification number for B2B invoicing compliance
    tax_id VARCHAR(50),

    -- Primary business industry for analytics and reporting
    industry VARCHAR(100),

    -- Total number of employees at the company (for plan sizing)
    employee_count INTEGER CHECK (employee_count > 0),

    -- Corporate headquarters address for contract purposes
    billing_address_line1 VARCHAR(255),
    billing_address_line2 VARCHAR(255),
    billing_city VARCHAR(100),
    billing_state VARCHAR(100),
    billing_postal_code VARCHAR(20),
    billing_country VARCHAR(100) DEFAULT 'USA',

    -- Primary contact person for the corporate account
    primary_contact_name VARCHAR(255) NOT NULL,
    primary_contact_email VARCHAR(255) NOT NULL,
    primary_contact_phone VARCHAR(50),

    -- Stripe Connect customer ID for payment processing
    stripe_customer_id VARCHAR(255),

    -- Payment terms in days (default Net-30 as per PRD)
    payment_terms_days INTEGER DEFAULT 30 CHECK (payment_terms_days >= 0),

    -- Enterprise procurement system integration identifiers
    sap_vendor_id VARCHAR(100),         -- SAP integration ID
    coupa_supplier_id VARCHAR(100),      -- Coupa integration ID
    -- Account status tracking
    is_active BOOLEAN DEFAULT true,

    -- Contract dates for subscription management
    contract_start_date DATE NOT NULL,
    contract_end_date DATE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on company name for search and autocomplete functionality
CREATE INDEX idx_corporates_company_name ON corporates(company_name);
-- Index on active status for filtering active corporate accounts
CREATE INDEX idx_corporates_is_active ON corporates(is_active);
-- Index on contract dates for renewal tracking queries
CREATE INDEX idx_corporates_contract_dates ON corporates(contract_start_date, contract_end_date);

-- Add table comment for documentation
COMMENT ON TABLE corporates IS 'B2B corporate accounts that purchase activity subscriptions for employees';


-- ==========================================================================
-- USERS TABLE
-- Unified user table supporting all four user roles: Parent, HR Admin,
-- Venue Admin, and Platform Admin with role-based access control
-- ==========================================================================
CREATE TABLE users (
    -- Primary identifier using UUID for security
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- User email address - used for login and notifications (must be unique)
    email VARCHAR(255) NOT NULL UNIQUE,

    -- Argon2/bcrypt hashed password - NULL for SSO-only users
    password_hash VARCHAR(255),

    -- User's role determining their access level and UI experience
    role user_role NOT NULL,

    -- User's display name shown in the UI
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,

    -- User's phone number for SMS/WhatsApp notifications
```

```
phone VARCHAR(50),

-- URL or path to user's profile avatar image
avatar_url VARCHAR(500),

-- Foreign key to corporate account (required for parents and HR admins)
corporate_id UUID REFERENCES corporates(id) ON DELETE SET NULL,

-- Foreign key to venue (required for venue admins, defined after venues table)
-- venue_id UUID REFERENCES venues(id) ON DELETE SET NULL,

-- Department within the corporate for HR reporting aggregation
department VARCHAR(100),

-- SSO configuration for corporate users
sso_provider VARCHAR(50),           -- 'saml' or 'oidc'
sso_subject_id VARCHAR(255),        -- Unique ID from SSO provider

-- Multi-factor authentication settings
mfa_enabled BOOLEAN DEFAULT false,
mfa_secret VARCHAR(255),            -- TOTP secret (encrypted at app level)

-- Account status flags
is_active BOOLEAN DEFAULT true,
is_email_verified BOOLEAN DEFAULT false,
email_verified_at TIMESTAMP WITH TIME ZONE,

-- Password reset token management
password_reset_token VARCHAR(255),
password_reset_expires_at TIMESTAMP WITH TIME ZONE,

-- Last login tracking for security and engagement metrics
last_login_at TIMESTAMP WITH TIME ZONE,
last_login_ip INET,

-- Notification preferences stored as flexible JSONB for extensibility
-- Structure: { "email": true, "sms": false, "push": true, "frequency": "immediate" }
notification_preferences JSONB DEFAULT '{
  "email": true,
  "sms": false,
  "whatsapp": false,
  "push": true,
  "frequency": "immediate"
}'::jsonb,

-- User's timezone for displaying local times
timezone VARCHAR(50) DEFAULT 'America/New_York',
```

```sql
    -- Preferred language for UI and notifications
    preferred_language VARCHAR(10) DEFAULT 'en',

    -- Remember me token for persistent login sessions (UI Spec line 42)
    remember_token VARCHAR(255),
    remember_token_expires_at TIMESTAMP WITH TIME ZONE,

    -- Coworker matching consent (PRD Feature 4 & UI Spec line 413)
    -- Allows children to be matched with coworkers' children for group activities
    allow_coworker_matching BOOLEAN DEFAULT false,
    coworker_matching_consent_date TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on email for login queries (already has unique constraint)
-- Index on role for filtering users by type
CREATE INDEX idx_users_role ON users(role);
-- Index on corporate_id for finding all users in a company
CREATE INDEX idx_users_corporate_id ON users(corporate_id);
-- Composite index for active users by role (common dashboard query)
CREATE INDEX idx_users_active_role ON users(is_active, role);
-- Index on SSO identifiers for SSO login lookup
CREATE INDEX idx_users_sso ON users(sso_provider, sso_subject_id) WHERE sso_provider IS NOT NULL;
-- GIN index on notification_preferences for JSONB queries
CREATE INDEX idx_users_notification_prefs ON users USING GIN (notification_preferences);

COMMENT ON TABLE users IS 'Unified user accounts supporting parents, HR admins, venue admins, and platform admins


-- ========================================================================================
-- PARENT LOCATION PREFERENCES TABLE
-- Stores multiple addresses (home, work, school) for geo-fence filtering
-- when recommending activities within 15-minute drive distance
-- ========================================================================================
CREATE TABLE user_locations (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the parent user (only parents need location preferences)
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Location type identifier
    location_type VARCHAR(20) NOT NULL CHECK (location_type IN ('home', 'work', 'school', 'other')),
    -- Human-readable label for the location
    label VARCHAR(100),

    -- Full address components for display
    address_line1 VARCHAR(255) NOT NULL,
    address_line2 VARCHAR(255),
    city VARCHAR(100) NOT NULL,
    state VARCHAR(100),
    postal_code VARCHAR(20) NOT NULL,
    country VARCHAR(100) DEFAULT 'USA',

    -- Geocoded coordinates for distance calculations (stored as separate fields)
    -- Note: For production, consider PostGIS GEOGRAPHY type for proper spatial queries
    latitude DECIMAL(10, 8),          -- Range: -90 to +90
    longitude DECIMAL(11, 8),         -- Range: -180 to +180

    -- Flag indicating if this is the primary address for search radius
    is_primary BOOLEAN DEFAULT false,

    -- Preferred search radius in minutes driving time (default 15 per PRD)
    default_search_radius_minutes INTEGER DEFAULT 15 CHECK (default_search_radius_minutes BETWEEN 5 AND 60)

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure only one location of each type per user
    CONSTRAINT unique_user_location_type UNIQUE (user_id, location_type)
);

-- Index on user_id for fetching all locations for a user
CREATE INDEX idx_user_locations_user_id ON user_locations(user_id);
-- Index on coordinates for proximity queries
CREATE INDEX idx_user_locations_coords ON user_locations(latitude, longitude) WHERE latitude IS NOT NULL;

COMMENT ON TABLE user_locations IS 'Parent address preferences for geo-fence filtering of activity recommendations';
```

2.4 Venue and Activity Tables

```
sql
```

```sql
-- ==============================================================================
-- VENUES TABLE
-- Activity providers (gymnasiums, art studios, STEM centers, etc.)
-- Subject to vetting and approval process before going live
-- ==============================================================================
CREATE TABLE venues (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Venue's business name as displayed to parents
    name VARCHAR(255) NOT NULL,

    -- URL-friendly slug for venue profile pages
    slug VARCHAR(255) NOT NULL UNIQUE,

    -- Brief tagline for venue cards (150 char limit per UI spec)
    short_description VARCHAR(150),

    -- Full venue description with amenities and philosophy
    full_description TEXT,

    -- Primary activity category this venue specializes in
    primary_category activity_category,

    -- Venue approval and visibility status
    status venue_status DEFAULT 'pending_approval',

    -- Physical address of the venue
    address_line1 VARCHAR(255) NOT NULL,
    address_line2 VARCHAR(255),
    city VARCHAR(100) NOT NULL,
    state VARCHAR(100) NOT NULL,
    postal_code VARCHAR(20) NOT NULL,
    country VARCHAR(100) DEFAULT 'USA',

    -- Geocoded coordinates for distance calculations
    latitude DECIMAL(10, 8) NOT NULL,
    longitude DECIMAL(11, 8) NOT NULL,

    -- Contact information
    contact_email VARCHAR(255) NOT NULL,
    contact_phone VARCHAR(50) NOT NULL,
    website_url VARCHAR(500),

    -- Business hours stored as JSONB for flexibility
    -- Structure: { "monday": {"open": "09:00", "close": "18:00"}, ... }
```

```sql
business_hours JSONB DEFAULT '{}'::jsonb,

-- Media assets
logo_url VARCHAR(500),
-- Array of image URLs for venue gallery
gallery_images JSONB DEFAULT '[]'::jsonb,

-- Insurance compliance (required per PRD Section 5.3)
insurance_provider VARCHAR(255),
insurance_policy_number VARCHAR(100),
insurance_coverage_amount DECIMAL(12, 2) CHECK (insurance_coverage_amount >= 1000000),
insurance_expiry_date DATE,
insurance_document_url VARCHAR(500),

-- Background check compliance tracking
background_check_completed BOOLEAN DEFAULT false,
background_check_date DATE,

-- Safety audit compliance
safety_audit_completed BOOLEAN DEFAULT false,
safety_audit_date DATE,
safety_audit_score INTEGER CHECK (safety_audit_score BETWEEN 0 AND 100),

-- Accessibility features available at the venue
accessibility_features JSONB DEFAULT '[]'::jsonb,

-- Banking information for payouts (encrypted at app level)
payout_bank_name VARCHAR(255),
payout_account_last_four VARCHAR(4),
stripe_connect_account_id VARCHAR(255),

-- Platform fee percentage for this venue (default 15-20% per PRD)
platform_fee_percentage DECIMAL(5, 2) DEFAULT 17.50 CHECK (platform_fee_percentage BETWEEN 0 AND 100),

-- Block booking configuration
-- Number of days in advance the platform pre-books slots
block_booking_advance_days INTEGER DEFAULT 30,
-- Hours before activity when unsold inventory is released back
inventory_release_hours INTEGER DEFAULT 72,

-- Current computed performance score (0-100, recalculated weekly)
current_performance_score DECIMAL(5, 2) DEFAULT 50.00 CHECK (current_performance_score BETWEEN 0 AND 1

-- Flag indicating if venue is approved for proactive curation (score ≥60)
is_curation_eligible BOOLEAN DEFAULT false,

-- Annual re-verification tracking
```

```sql
    last_verification_date DATE,
    next_verification_due DATE,

    -- Admin user who manages this venue
    primary_admin_user_id UUID,

    -- Rejection feedback if application was denied
    rejection_reason TEXT,
    rejection_date TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    approved_at TIMESTAMP WITH TIME ZONE
);

-- Index on slug for URL lookups
CREATE INDEX idx_venues_slug ON venues(slug);
-- Index on status for filtering pending/active venues
CREATE INDEX idx_venues_status ON venues(status);
-- Spatial index on coordinates for proximity searches
CREATE INDEX idx_venues_coordinates ON venues(latitude, longitude);
-- Index on performance score for curation filtering (exclude <60)
CREATE INDEX idx_venues_performance ON venues(current_performance_score) WHERE status = 'active';
-- Index on insurance expiry for compliance alerts
CREATE INDEX idx_venues_insurance_expiry ON venues(insurance_expiry_date);
-- Index on category for filtering
CREATE INDEX idx_venues_category ON venues(primary_category);
-- GIN index on accessibility features for JSONB array queries
CREATE INDEX idx_venues_accessibility ON venues USING GIN (accessibility_features);

COMMENT ON TABLE venues IS 'Activity provider venues subject to vetting, approval, and performance scoring';


-- Now add the venue_id foreign key to users table
ALTER TABLE users ADD COLUMN venue_id UUID REFERENCES venues(id) ON DELETE SET NULL;
CREATE INDEX idx_users_venue_id ON users(venue_id) WHERE venue_id IS NOT NULL;


-- ===========================================================================
-- VENUE PERFORMANCE SCORES HISTORY TABLE
-- Weekly snapshots of venue performance metrics for trend analysis
-- Score components weighted per PRD: Parent feedback 40%, Repeat booking 25%,
-- Cancellation rate 20% (negative), No-show rate 15%
-- ===========================================================================
CREATE TABLE venue_performance_scores (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the venue being scored
    venue_id UUID NOT NULL REFERENCES venues(id) ON DELETE CASCADE,

    -- Week start date for this score snapshot (always Monday)
    score_week_start DATE NOT NULL,

    -- Individual component scores (0-100 scale)
    parent_feedback_score DECIMAL(5, 2) CHECK (parent_feedback_score BETWEEN 0 AND 100),
    repeat_booking_score DECIMAL(5, 2) CHECK (repeat_booking_score BETWEEN 0 AND 100),
    cancellation_score DECIMAL(5, 2) CHECK (cancellation_score BETWEEN 0 AND 100),
    no_show_score DECIMAL(5, 2) CHECK (no_show_score BETWEEN 0 AND 100),

    -- Raw metrics used in calculation (for transparency and debugging)
    total_bookings INTEGER DEFAULT 0,
    total_reviews INTEGER DEFAULT 0,
    average_rating DECIMAL(3, 2),
    repeat_booking_rate DECIMAL(5, 2),
    venue_cancellation_rate DECIMAL(5, 2),
    parent_no_show_rate DECIMAL(5, 2),

    -- Final weighted composite score
    composite_score DECIMAL(5, 2) NOT NULL CHECK (composite_score BETWEEN 0 AND 100),

    -- Score change from previous week
    score_change DECIMAL(5, 2),

    -- AI-generated improvement suggestions stored as JSONB array
    improvement_suggestions JSONB DEFAULT '[]'::jsonb,

    -- Timestamp when this score was calculated
    calculated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure one score per venue per week
    CONSTRAINT unique_venue_weekly_score UNIQUE (venue_id, score_week_start)
);

-- Index for retrieving score history for a venue
CREATE INDEX idx_venue_scores_venue_id ON venue_performance_scores(venue_id);
-- Index for retrieving scores by week (for platform-wide reporting)
CREATE INDEX idx_venue_scores_week ON venue_performance_scores(score_week_start);
-- Composite index for querying recent scores
CREATE INDEX idx_venue_scores_recent ON venue_performance_scores(venue_id, score_week_start DESC);

COMMENT ON TABLE venue_performance_scores IS 'Weekly venue performance score snapshots with component breakd
```

```sql
-- ============================================================================
-- VENUE PAYOUTS TABLE
-- Tracks revenue and payouts to venues (UI Spec lines 591-593, 736)
-- Supports the platform fee model from PRD Section 2.1
-- ============================================================================
CREATE TABLE venue_payouts (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the venue
    venue_id UUID NOT NULL REFERENCES venues(id) ON DELETE CASCADE,

    -- Payout period
    payout_period_start DATE NOT NULL,
    payout_period_end DATE NOT NULL,

    -- Booking metrics for this period
    total_bookings INTEGER DEFAULT 0,
    total_attendees INTEGER DEFAULT 0,

    -- Revenue calculations
    gross_revenue DECIMAL(12, 2) NOT NULL DEFAULT 0 CHECK (gross_revenue >= 0),
    platform_fee_percentage DECIMAL(5, 2) NOT NULL,
    platform_fee_amount DECIMAL(12, 2) NOT NULL DEFAULT 0 CHECK (platform_fee_amount >= 0),
    net_payout DECIMAL(12, 2) NOT NULL DEFAULT 0 CHECK (net_payout >= 0),

    -- Adjustments (refunds, disputes, etc.)
    adjustments_amount DECIMAL(12, 2) DEFAULT 0,
    adjustments_notes TEXT,

    -- Payout status
    status VARCHAR(20) DEFAULT 'pending'
        CHECK (status IN ('pending', 'processing', 'paid', 'failed', 'on_hold')),

    -- Stripe payout tracking
    stripe_payout_id VARCHAR(255),
    stripe_transfer_id VARCHAR(255),

    -- Payment details
    paid_at TIMESTAMP WITH TIME ZONE,
    payment_method VARCHAR(50),  -- 'bank_transfer', 'stripe_connect'
    payment_reference VARCHAR(255),

    -- Failure tracking
    failure_reason TEXT,
    retry_count INTEGER DEFAULT 0,
    -- Invoice/statement generation
    statement_url VARCHAR(500),
    statement_generated_at TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure one payout record per venue per period
    CONSTRAINT unique_venue_payout_period UNIQUE (venue_id, payout_period_start)
);

-- Index on venue_id for payout history
CREATE INDEX idx_venue_payouts_venue ON venue_payouts(venue_id);
-- Index on status for processing queue
CREATE INDEX idx_venue_payouts_status ON venue_payouts(status) WHERE status IN ('pending', 'processing');
-- Index on period for reporting
CREATE INDEX idx_venue_payouts_period ON venue_payouts(payout_period_start, payout_period_end);

COMMENT ON TABLE venue_payouts IS 'Venue revenue tracking and payout management';


-- ==========================================================================
-- VENUE INTEGRATIONS TABLE
-- Tracks MCP (Model Context Protocol) integrations per venue (PRD Feature 5)
-- Different venues may use different booking systems
-- ==========================================================================
CREATE TABLE venue_integrations (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the venue
    venue_id UUID NOT NULL REFERENCES venues(id) ON DELETE CASCADE,

    -- Integration type/provider
    integration_type VARCHAR(50) NOT NULL,  -- 'mindbody', 'acuity', 'calendly', 'custom_api', 'manual'

    -- Integration display name
    display_name VARCHAR(100),

    -- API configuration (encrypted at app level)
    -- Structure varies by integration type
    api_config JSONB DEFAULT '{}'::jsonb,

    -- Authentication credentials reference (stored in secrets manager)
    credentials_secret_id VARCHAR(255),
    -- Webhook configuration for real-time updates
    webhook_url VARCHAR(500),
    webhook_secret VARCHAR(255),

    -- Sync configuration
    sync_enabled BOOLEAN DEFAULT true,
    sync_frequency_minutes INTEGER DEFAULT 60,  -- How often to sync inventory
    last_sync_at TIMESTAMP WITH TIME ZONE,
    last_sync_status VARCHAR(20),  -- 'success', 'partial', 'failed'
    last_sync_error TEXT,

    -- Inventory sync tracking (PRD Feature 5 - Inventory Sync Watchdog)
    last_inventory_check_at TIMESTAMP WITH TIME ZONE,
    inventory_discrepancy_count INTEGER DEFAULT 0,

    -- Integration health
    is_active BOOLEAN DEFAULT true,
    health_status VARCHAR(20) DEFAULT 'unknown',  -- 'healthy', 'degraded', 'down', 'unknown'
    health_check_at TIMESTAMP WITH TIME ZONE,

    -- Fallback configuration
    fallback_to_email BOOLEAN DEFAULT true,  -- Use email if integration fails
    fallback_email VARCHAR(255),

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- One integration type per venue
    CONSTRAINT unique_venue_integration UNIQUE (venue_id, integration_type)
);

-- Index on venue_id
CREATE INDEX idx_venue_integrations_venue ON venue_integrations(venue_id);
-- Index on active integrations needing sync
CREATE INDEX idx_venue_integrations_sync ON venue_integrations(last_sync_at, sync_frequency_minutes)
    WHERE is_active = true AND sync_enabled = true;
-- Index on health status for monitoring
CREATE INDEX idx_venue_integrations_health ON venue_integrations(health_status)
    WHERE is_active = true;

COMMENT ON TABLE venue_integrations IS 'MCP integration configurations per venue for booking system sync';


-- ==========================================================================
-- ACTIVITIES TABLE
-- Individual activity types offered by venues (e.g., "Robotics Workshop",
-- "Watercolor Painting Class"). Each activity can have multiple scheduled sessions.
-- ==========================================================================
CREATE TABLE activities (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the venue offering this activity
    venue_id UUID NOT NULL REFERENCES venues(id) ON DELETE CASCADE,

    -- Activity name as displayed to parents
    name VARCHAR(255) NOT NULL,

    -- URL-friendly slug for activity detail pages
    slug VARCHAR(255) NOT NULL,

    -- Primary category for filtering and recommendations
    category activity_category NOT NULL,

    -- Brief description for activity cards (150 char limit)
    short_description VARCHAR(150),

    -- Full activity description with learning outcomes
    full_description TEXT,

    -- Learning outcomes as JSONB array for "What your child will learn" section
    -- Structure: ["Learn basic coding concepts", "Build a working robot", ...]
    learning_outcomes JSONB DEFAULT '[]'::jsonb,

    -- Items participants should bring
    -- Structure: ["Comfortable clothes", "Water bottle", ...]
    what_to_bring JSONB DEFAULT '[]'::jsonb,

    -- Age range constraints for eligibility
    min_age INTEGER NOT NULL CHECK (min_age >= 0 AND min_age <= 18),
    max_age INTEGER NOT NULL CHECK (max_age >= 0 AND max_age <= 18),

    -- Duration of each session in minutes
    duration_minutes INTEGER NOT NULL CHECK (duration_minutes > 0),

    -- Credits required to book this activity (deducted at booking time)
    credits_required INTEGER NOT NULL CHECK (credits_required > 0),

    -- Maximum participants per session
    capacity_per_session INTEGER NOT NULL CHECK (capacity_per_session > 0),

    -- Media assets for activity display
```

```sql
    primary_image_url VARCHAR(500),
    gallery_images JSONB DEFAULT '[]'::jsonb,

    -- Activity characteristics for filtering (stored as JSONB for flexibility)
    -- Structure: { "indoor": true, "messy": false, "competitive": false, ... }
    activity_tags JSONB DEFAULT '{}'::jsonb,

    -- Accessibility features specific to this activity
    accessibility_features JSONB DEFAULT '[]'::jsonb,

    -- Whether parent must attend with child
    parent_attendance_required BOOLEAN DEFAULT false,

    -- Skill level requirement
    skill_level VARCHAR(20) DEFAULT 'beginner' CHECK (skill_level IN ('beginner', 'intermediate', 'advanced')),

    -- Prerequisites text (free-form)
    prerequisites TEXT,

    -- Activity visibility and status
    is_active BOOLEAN DEFAULT true,
    is_published BOOLEAN DEFAULT false,

    -- ==========================================================================
    -- PINECONE INTEGRATION FIELDS
    -- Vector storage is in Pinecone; PostgreSQL only tracks sync status
    -- ==========================================================================

    -- Pinecone vector ID for this activity's description embedding
    -- Format: "act_{uuid}" - stored in 'activity-embeddings' namespace
    pinecone_vector_id VARCHAR(255),

    -- Timestamp when embedding was last synced to Pinecone
    -- Used by sync job to detect stale vectors needing re-embedding
    pinecone_synced_at TIMESTAMP WITH TIME ZONE,

    -- Flag indicating if activity content changed since last Pinecone sync
    -- Set to TRUE on any update to name, description, tags, or category
    -- Sync job resets to FALSE after successful embedding update
    pinecone_sync_required BOOLEAN DEFAULT true,

    -- Embedding model version used (for bulk re-embedding on model upgrades)
    -- Example: "text-embedding-3-small-v1"
    embedding_model_version VARCHAR(50),

    -- ==========================================================================
    -- DENORMALIZED FIELDS (Computed, not for Pinecone)
    -- ==========================================================================

    -- Average rating computed from reviews (denormalized for performance)
    average_rating DECIMAL(3, 2) DEFAULT 0.00 CHECK (average_rating BETWEEN 0 AND 5),
    total_reviews INTEGER DEFAULT 0,

    -- Total bookings count (denormalized for sorting)
    total_bookings INTEGER DEFAULT 0,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    published_at TIMESTAMP WITH TIME ZONE
);

-- Ensure unique slug per venue
ALTER TABLE activities ADD CONSTRAINT unique_activity_slug_per_venue UNIQUE (venue_id, slug);

-- Index on venue_id for retrieving all activities at a venue
CREATE INDEX idx_activities_venue_id ON activities(venue_id);
-- Index on category for filter queries
CREATE INDEX idx_activities_category ON activities(category);
-- Index on age range for eligibility filtering
CREATE INDEX idx_activities_age_range ON activities(min_age, max_age);
-- Index on credits for sorting by price
CREATE INDEX idx_activities_credits ON activities(credits_required);
-- Index on active and published activities
CREATE INDEX idx_activities_active ON activities(is_active, is_published) WHERE is_active = true AND is_published = t
-- GIN index on activity_tags for JSONB queries (e.g., finding non-messy activities)
CREATE INDEX idx_activities_tags ON activities USING GIN (activity_tags);
-- Index on average rating for sorting
CREATE INDEX idx_activities_rating ON activities(average_rating DESC);
-- Index for Pinecone sync job (find activities needing re-embedding)
CREATE INDEX idx_activities_pinecone_sync ON activities(pinecone_sync_required)
    WHERE pinecone_sync_required = true AND is_active = true;
-- Full-text search index on name and description (FALLBACK when Pinecone unavailable)
-- Per PRD Section 6.3: Fallback to PostgreSQL-based filtering with degraded personalization
CREATE INDEX idx_activities_search ON activities USING GIN (
    to_tsvector('english', name || ' ' || COALESCE(short_description, '') || ' ' || COALESCE(full_description, ''))
);

COMMENT ON TABLE activities IS 'Activity types offered by venues with metadata for search etc';



-- ==========================================================================
-- ACTIVITY RECURRING SCHEDULES TABLE
-- Defines recurring patterns for activities (UI Spec lines 639-644)
-- Used to auto-generate activity_sessions based on pattern rules
-- ==========================================================================
CREATE TABLE activity_recurring_schedules (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the activity this schedule is for
    activity_id UUID NOT NULL REFERENCES activities(id) ON DELETE CASCADE,

    -- Schedule name for venue admin reference
    schedule_name VARCHAR(100),

    -- Days of week this activity runs (stored as array)
    -- Structure: [0, 1, 2, 3, 4, 5, 6] where 0=Sunday, 6=Saturday
    days_of_week JSONB NOT NULL DEFAULT '[]'::jsonb,

    -- Start time for sessions on these days
    start_time TIME NOT NULL,

    -- Duration in minutes (to calculate end_time)
    duration_minutes INTEGER NOT NULL CHECK (duration_minutes > 0),

    -- Capacity per session
    capacity INTEGER NOT NULL CHECK (capacity > 0),

    -- Date range for this recurring schedule
    effective_from DATE NOT NULL,
    effective_until DATE,  -- NULL means ongoing

    -- Block booking allocation per session
    platform_allocated_spots INTEGER DEFAULT 0,

    -- Instructor assignment (if consistent)
    default_instructor_name VARCHAR(255),

    -- Is this schedule currently active?
    is_active BOOLEAN DEFAULT true,

    -- Last date sessions were generated up to
    sessions_generated_until DATE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on activity_id for retrieving schedules
```

```sql
CREATE INDEX idx_recurring_schedules_activity ON activity_recurring_schedules(activity_id);
-- Index on active schedules for session generation job
CREATE INDEX idx_recurring_schedules_active ON activity_recurring_schedules(is_active, effective_from)
  WHERE is_active = true;
-- GIN index on days_of_week for day filtering
CREATE INDEX idx_recurring_schedules_days ON activity_recurring_schedules USING GIN (days_of_week);

COMMENT ON TABLE activity_recurring_schedules IS 'Recurring schedule patterns for auto-generating activity sessions';



-- ============================================================================
-- ACTIVITY SESSIONS TABLE
-- Specific scheduled instances of activities (time slots)
-- Supports the block booking model where platform pre-books slots 30 days ahead
-- ============================================================================
CREATE TABLE activity_sessions (
  -- Primary identifier
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

  -- Reference to the activity type
  activity_id UUID NOT NULL REFERENCES activities(id) ON DELETE CASCADE,

  -- Reference to recurring schedule (if generated from a pattern)
  recurring_schedule_id UUID REFERENCES activity_recurring_schedules(id) ON DELETE SET NULL,

  -- Session date and time
  session_date DATE NOT NULL,
  start_time TIME NOT NULL,
  end_time TIME NOT NULL,

  -- Timezone for this session (inherited from venue but can be overridden)
  timezone VARCHAR(50) DEFAULT 'America/New_York',

  -- Time of day classification for filtering (UI Spec line 196)
  -- Computed based on start_time: morning (<12), afternoon (12-17), evening (>17)
  time_of_day time_of_day,

  -- Capacity management
  total_capacity INTEGER NOT NULL CHECK (total_capacity > 0),
  booked_count INTEGER DEFAULT 0 CHECK (booked_count >= 0),

  -- Block booking allocation (spots pre-booked by platform)
  platform_allocated_spots INTEGER DEFAULT 0,

  -- Remaining spots for waitlist threshold
  -- Computed as: total_capacity - booked_count
  -- When this reaches 0, new bookings go to waitlist
```

```sql
    -- Session status
    is_cancelled BOOLEAN DEFAULT false,
    cancellation_reason TEXT,
    cancelled_at TIMESTAMP WITH TIME ZONE,
    cancelled_by UUID REFERENCES users(id),

    -- Session completion tracking
    is_completed BOOLEAN DEFAULT false,
    completed_at TIMESTAMP WITH TIME ZONE,

    -- Instructor assignment (optional tracking)
    instructor_name VARCHAR(255),
    instructor_notes TEXT,

    -- Special notes for this session
    session_notes TEXT,

    -- Release tracking (72-hour release clause)
    inventory_released BOOLEAN DEFAULT false,
    inventory_released_at TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure end time is after start time
    CONSTRAINT valid_session_times CHECK (end_time > start_time),
    -- Ensure booked count doesn't exceed capacity
    CONSTRAINT valid_booking_count CHECK (booked_count <= total_capacity)
);

-- Index on activity_id for retrieving all sessions of an activity
CREATE INDEX idx_sessions_activity_id ON activity_sessions(activity_id);
-- Index on session_date for date range queries
CREATE INDEX idx_sessions_date ON activity_sessions(session_date);
-- Composite index for finding available sessions (most common query)
CREATE INDEX idx_sessions_availability ON activity_sessions(activity_id, session_date, is_cancelled)
    WHERE is_cancelled = false;
-- Index for finding sessions with available spots
CREATE INDEX idx_sessions_available_spots ON activity_sessions(session_date, booked_count, total_capacity)
    WHERE is_cancelled = false AND booked_count < total_capacity;
-- Index on cancelled status for reporting
CREATE INDEX idx_sessions_cancelled ON activity_sessions(is_cancelled) WHERE is_cancelled = true;
-- Index on time_of_day for filtering (UI Spec line 196)
CREATE INDEX idx_sessions_time_of_day ON activity_sessions(time_of_day, session_date)
    WHERE is_cancelled = false;
```

```
COMMENT ON TABLE activity_sessions IS 'Scheduled time slots for activities with capacity and booking management';
```

2.5 Children and Profile Tables

```
sql
```

```sql
-- ===============================================================================
-- CHILDREN TABLE
-- Child profiles registered by parents for activity matching
-- Subject to COPPA/GDPR-K compliance with strict data isolation
-- ===============================================================================
CREATE TABLE children (
    -- Primary identifier (never exposed externally, use initials for privacy)
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the parent user who registered this child
    parent_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Child's first name (no last name stored for privacy)
    first_name VARCHAR(100) NOT NULL,

    -- Date of birth for age calculation and eligibility checks
    date_of_birth DATE NOT NULL,

    -- Optional gender for personalization (not required)
    gender VARCHAR(20) CHECK (gender IN ('boy', 'girl', 'non_binary', 'prefer_not_to_say')),

    -- Profile photo URL (parent-uploaded)
    avatar_url VARCHAR(500),

    -- Onboarding quiz responses (3-question quiz per PRD)
    energy_level energy_level,
    social_preference social_preference,

    -- Interest categories selected during onboarding (up to 3)
    -- Stored as JSONB array: ["stem", "arts_crafts", "building"]
    interests JSONB DEFAULT '[]'::jsonb,

    -- Activities to avoid (parent-specified constraints)
    -- Structure: ["competitive", "messy", "loud", "physical_contact"]
    activities_to_avoid JSONB DEFAULT '[]'::jsonb,

    -- Medical and allergy notes (shared with venues for safety)
    allergies_medical_notes TEXT,

    -- Accessibility needs
    -- Structure: ["wheelchair_accessible", "sensory_friendly", ...]
    accessibility_needs JSONB DEFAULT '[]'::jsonb,

    -- ===========================================================================
    -- PINECONE INTEGRATION FIELDS
    -- Child preference vectors stored in Pinecone 'child-preferences' namespace
-- Used for semantic matching with activities and sibling compatibility
-- ==========================================================================

-- Pinecone vector ID for this child's preference embedding
-- Format: "child_{uuid}" - combines quiz answers + feedback history
pinecone_vector_id VARCHAR(255),

-- Timestamp when preference vector was last synced to Pinecone
pinecone_synced_at TIMESTAMP WITH TIME ZONE,

-- Flag indicating preferences changed since last Pinecone sync
-- Set TRUE when: quiz answers change, new feedback received, interests updated
pinecone_sync_required BOOLEAN DEFAULT true,

-- Embedding model version used for this child's vector
embedding_model_version VARCHAR(50),

-- Count of feedback events incorporated into current vector
-- Helps track vector evolution over time
feedback_incorporated_count INTEGER DEFAULT 0,

-- ==========================================================================
-- ACTIVITY TRACKING
-- ==========================================================================

-- Denormalized activity count for this month (for dashboard display)
activities_this_month INTEGER DEFAULT 0,

-- ==========================================================================
-- COPPA COMPLIANCE FIELDS
-- ==========================================================================

-- Parental consent tracking (COPPA compliance)
parental_consent_given BOOLEAN DEFAULT false,
parental_consent_date TIMESTAMP WITH TIME ZONE,
parental_consent_ip INET,

-- Soft delete support for data retention compliance
is_active BOOLEAN DEFAULT true,
deactivated_at TIMESTAMP WITH TIME ZONE,

-- Data deletion request tracking (GDPR-K right to deletion)
-- NOTE: When deletion occurs, must also delete from Pinecone
deletion_requested BOOLEAN DEFAULT false,
deletion_requested_at TIMESTAMP WITH TIME ZONE,
deletion_scheduled_for DATE,
    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on parent_id for retrieving all children of a parent
CREATE INDEX idx_children_parent_id ON children(parent_id);
-- Index on date_of_birth for age calculation queries
CREATE INDEX idx_children_dob ON children(date_of_birth);
-- Index on active children
CREATE INDEX idx_children_active ON children(parent_id, is_active) WHERE is_active = true;
-- GIN index on interests for filtering children by interest
CREATE INDEX idx_children_interests ON children USING GIN (interests);
-- Index for deletion queue processing
CREATE INDEX idx_children_deletion_pending ON children(deletion_scheduled_for)
    WHERE deletion_requested = true;
-- Index for Pinecone sync job (find children needing vector update)
CREATE INDEX idx_children_pinecone_sync ON children(pinecone_sync_required)
    WHERE pinecone_sync_required = true AND is_active = true;

COMMENT ON TABLE children IS 'Child profiles for activity matching, subject to COPPA/GDPR-K compliance. Preferenc


-- ==========================================================================
-- CHILD FAVORITES TABLE
-- Activities favorited/saved by parents for quick access
-- ==========================================================================
CREATE TABLE child_favorites (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the child this favorite is for
    child_id UUID NOT NULL REFERENCES children(id) ON DELETE CASCADE,

    -- Reference to the favorited activity
    activity_id UUID NOT NULL REFERENCES activities(id) ON DELETE CASCADE,

    -- Timestamp when favorited
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure a child can only favorite an activity once
    CONSTRAINT unique_child_favorite UNIQUE (child_id, activity_id)
);

-- Index for retrieving favorites for a child
CREATE INDEX idx_favorites_child_id ON child_favorites(child_id);
```

```
COMMENT ON TABLE child_favorites IS 'Parent-saved favorite activities for quick access';
```

2.6 Subscription and Credit Management Tables

```
sql
```

```sql
-- ===========================================================================
-- CORPORATE SUBSCRIPTIONS TABLE
-- Subscription periods for corporate accounts with credit allocation rules
-- ===========================================================================
CREATE TABLE corporate_subscriptions (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the corporate account
    corporate_id UUID NOT NULL REFERENCES corporates(id) ON DELETE CASCADE,

    -- Subscription tier determining features and credit limits
    tier subscription_tier NOT NULL,

    -- Number of employees covered by this subscription
    covered_employees INTEGER NOT NULL CHECK (covered_employees > 0),

    -- Credits allocated per employee per month
    credits_per_employee INTEGER NOT NULL CHECK (credits_per_employee > 0),

    -- Monthly subscription cost (for invoicing)
    monthly_cost DECIMAL(12, 2) NOT NULL CHECK (monthly_cost >= 0),

    -- Billing cycle day of month (1-28 to avoid month-end issues)
    billing_cycle_day INTEGER DEFAULT 1 CHECK (billing_cycle_day BETWEEN 1 AND 28),

    -- Subscription period
    start_date DATE NOT NULL,
    end_date DATE,

    -- Auto-renewal configuration
    auto_renew BOOLEAN DEFAULT true,

    -- Premium venue access flag (for premium/enterprise tiers)
    premium_venue_access BOOLEAN DEFAULT false,

    -- Custom terms for enterprise accounts (stored as JSONB)
    custom_terms JSONB,

    -- Subscription status
    is_active BOOLEAN DEFAULT true,
    cancelled_at TIMESTAMP WITH TIME ZONE,
    cancellation_reason TEXT,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on corporate_id for retrieving subscription history
CREATE INDEX idx_subscriptions_corporate_id ON corporate_subscriptions(corporate_id);
-- Index on active subscriptions
CREATE INDEX idx_subscriptions_active ON corporate_subscriptions(corporate_id, is_active)
    WHERE is_active = true;
-- Index on end dates for renewal processing
CREATE INDEX idx_subscriptions_end_date ON corporate_subscriptions(end_date);

COMMENT ON TABLE corporate_subscriptions IS 'Corporate subscription periods with credit allocation rules';


-- ============================================================================
-- EMPLOYEE INVITATIONS TABLE
-- Tracks HR-initiated invitations to employees (UI Spec lines 506-513)
-- Supports bulk CSV upload and manual invitation workflows
-- ============================================================================
CREATE TABLE employee_invitations (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the corporate account
    corporate_id UUID NOT NULL REFERENCES corporates(id) ON DELETE CASCADE,

    -- Reference to the HR admin who sent the invitation
    invited_by UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Invitee details
    email VARCHAR(255) NOT NULL,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    department VARCHAR(100),

    -- Credits to allocate once they register
    credits_to_allocate INTEGER NOT NULL CHECK (credits_to_allocate > 0),

    -- Invitation token for registration link
    invitation_token VARCHAR(255) NOT NULL UNIQUE,

    -- Invitation status tracking
    status invitation_status DEFAULT 'pending',

    -- Token expiration (typically 7-14 days)
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,
```

```sql
    -- If accepted, reference to the created user account
    accepted_user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    accepted_at TIMESTAMP WITH TIME ZONE,

    -- Resend tracking
    resend_count INTEGER DEFAULT 0,
    last_resent_at TIMESTAMP WITH TIME ZONE,

    -- Revocation details
    revoked_at TIMESTAMP WITH TIME ZONE,
    revoked_by UUID REFERENCES users(id),
    revoke_reason TEXT,

    -- Batch import tracking (for CSV uploads)
    batch_import_id UUID,  -- Groups invitations from same CSV upload

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on corporate_id for listing all invitations
CREATE INDEX idx_invitations_corporate_id ON employee_invitations(corporate_id);
-- Index on email for duplicate checking
CREATE INDEX idx_invitations_email ON employee_invitations(corporate_id, email);
-- Index on status for filtering pending invitations
CREATE INDEX idx_invitations_status ON employee_invitations(status);
-- Index on token for registration lookup
CREATE INDEX idx_invitations_token ON employee_invitations(invitation_token) WHERE status = 'pending';
-- Index on expiration for cleanup job
CREATE INDEX idx_invitations_expiry ON employee_invitations(expires_at) WHERE status = 'pending';
-- Index on batch for grouping CSV imports
CREATE INDEX idx_invitations_batch ON employee_invitations(batch_import_id) WHERE batch_import_id IS NOT NUL

COMMENT ON TABLE employee_invitations IS 'HR-initiated employee invitations with token-based registration';


-- ================================================================================
-- USER CREDIT BALANCES TABLE
-- Current credit balance for each parent user (denormalized from ledger)
-- Reset monthly based on corporate subscription billing cycle
-- ================================================================================
CREATE TABLE user_credit_balances (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the parent user
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Current billing period (month start date)
    billing_period_start DATE NOT NULL,
    billing_period_end DATE NOT NULL,

    -- Credit allocation for this period
    credits_allocated INTEGER NOT NULL CHECK (credits_allocated >= 0),

    -- Current available balance (allocated - used + refunded)
    credits_available INTEGER NOT NULL CHECK (credits_available >= 0),

    -- Credits already used this period
    credits_used INTEGER DEFAULT 0 CHECK (credits_used >= 0),

    -- Credits refunded this period
    credits_refunded INTEGER DEFAULT 0 CHECK (credits_refunded >= 0),

    -- Credits forfeited (no-shows and late cancellations)
    credits_forfeited INTEGER DEFAULT 0 CHECK (credits_forfeited >= 0),

    -- Timestamp of last credit refresh
    last_refresh_at TIMESTAMP WITH TIME ZONE,

    -- Expiry warning sent flag (7 days before period end)
    expiry_warning_sent BOOLEAN DEFAULT false,
    expiry_warning_sent_at TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure one active balance record per user per billing period
    CONSTRAINT unique_user_billing_period UNIQUE (user_id, billing_period_start)
);

-- Index on user_id for retrieving current balance
CREATE INDEX idx_credit_balances_user_id ON user_credit_balances(user_id);
-- Index for finding expiring balances (7-day warning)
CREATE INDEX idx_credit_balances_expiry ON user_credit_balances(billing_period_end, expiry_warning_sent)
    WHERE expiry_warning_sent = false;

COMMENT ON TABLE user_credit_balances IS 'Current credit balance per parent per billing period';


-- ================================================================================
-- CREDIT LEDGER TABLE
```

```sql
-- Immutable transaction log of all credit movements for audit trail
-- Supports the financial operations described in PRD Section 2.2
-- ==========================================================================
CREATE TABLE credit_ledger (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the user whose credits are affected
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Reference to the balance record being modified
    balance_id UUID NOT NULL REFERENCES user_credit_balances(id) ON DELETE CASCADE,

    -- Type of credit transaction
    transaction_type credit_transaction_type NOT NULL,

    -- Credit amount (positive for credits in, negative for credits out)
    amount INTEGER NOT NULL,

    -- Running balance after this transaction
    balance_after INTEGER NOT NULL,

    -- Optional reference to related booking (for booking debits/refunds)
    booking_id UUID,  -- FK added after bookings table creation

    -- Description of the transaction
    description TEXT,

    -- Reference to admin who made adjustment (for manual adjustments)
    adjusted_by UUID REFERENCES users(id),

    -- Idempotency key to prevent duplicate transactions
    idempotency_key VARCHAR(255),

    -- Transaction timestamp (immutable)
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure idempotency key is unique if provided
    CONSTRAINT unique_idempotency_key UNIQUE (idempotency_key)
);

-- Index on user_id for retrieving transaction history
CREATE INDEX idx_credit_ledger_user_id ON credit_ledger(user_id);
-- Index on balance_id for period-specific queries
CREATE INDEX idx_credit_ledger_balance_id ON credit_ledger(balance_id);
-- Index on booking_id for finding credit transactions for a booking
CREATE INDEX idx_credit_ledger_booking_id ON credit_ledger(booking_id) WHERE booking_id IS NOT NULL;
```

```sql
-- Index on transaction type for reporting
CREATE INDEX idx_credit_ledger_type ON credit_ledger(transaction_type);
-- Index on created_at for date range queries
CREATE INDEX idx_credit_ledger_created_at ON credit_ledger(created_at);

COMMENT ON TABLE credit_ledger IS 'Immutable audit log of all credit transactions';
```

2.7 Booking and Waitlist Tables

```
sql
```

```sql
-- ===============================================================================
-- BOOKINGS TABLE
-- Activity reservations made by parents for their children
-- Core transaction table with status tracking through complete lifecycle
-- ===============================================================================
CREATE TABLE bookings (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the booked activity session
    session_id UUID NOT NULL REFERENCES activity_sessions(id) ON DELETE RESTRICT,

    -- Reference to the child attending (for single-child bookings)
    child_id UUID NOT NULL REFERENCES children(id) ON DELETE RESTRICT,

    -- Reference to the parent who made the booking
    booked_by UUID NOT NULL REFERENCES users(id) ON DELETE RESTRICT,

    -- Booking status tracking lifecycle
    status booking_status DEFAULT 'pending',

    -- Credits charged for this booking
    credits_charged INTEGER NOT NULL CHECK (credits_charged > 0),

    -- Group booking reference (links sibling bookings together)
    -- All bookings made in a single group transaction share this ID
    group_booking_id UUID,

    -- Confirmation details
    confirmation_code VARCHAR(20) NOT NULL UNIQUE,
    confirmed_at TIMESTAMP WITH TIME ZONE,

    -- Cancellation details (if applicable)
    cancelled_at TIMESTAMP WITH TIME ZONE,
    cancellation_reason cancellation_reason_type,  -- Structured reason (UI Spec line 330)
    cancellation_notes TEXT,                       -- Additional details if reason is 'other'

    -- Refund tracking
    refund_eligible BOOLEAN,
    refund_processed BOOLEAN DEFAULT false,
    refund_processed_at TIMESTAMP WITH TIME ZONE,
    credits_refunded INTEGER DEFAULT 0,

    -- Attendance tracking (marked by venue)
    attendance_marked BOOLEAN DEFAULT false,
    attendance_marked_at TIMESTAMP WITH TIME ZONE,
    attendance_marked_by UUID REFERENCES users(id),

    -- Special notes from parent (allergies reminder, etc.)
    parent_notes TEXT,

    -- Venue-side notes for this booking
    venue_notes TEXT,

    -- AI curation tracking (was this from a suggestion?)
    from_ai_suggestion BOOLEAN DEFAULT false,
    ai_suggestion_id UUID,  -- FK added after ai_suggestions table

    -- Source tracking for analytics
    booking_source VARCHAR(50) DEFAULT 'browse',  -- 'browse', 'suggestion', 'search', 'waitlist', 'rebook'

    -- Reference to original booking if this is a "Book Again" (UI Spec line 311)
    original_booking_id UUID REFERENCES bookings(id) ON DELETE SET NULL,

    -- Calendar event tracking
    calendar_event_sent BOOLEAN DEFAULT false,

    -- Reminder notifications sent
    reminder_24h_sent BOOLEAN DEFAULT false,
    reminder_1h_sent BOOLEAN DEFAULT false,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Row-level locking version for optimistic concurrency control (per PRD Section 8)
    version INTEGER DEFAULT 1
);

-- Index on session_id for finding all bookings for a session (attendance list)
CREATE INDEX idx_bookings_session_id ON bookings(session_id);
-- Index on child_id for finding all bookings for a child
CREATE INDEX idx_bookings_child_id ON bookings(child_id);
-- Index on booked_by for parent's booking history
CREATE INDEX idx_bookings_booked_by ON bookings(booked_by);
-- Index on status for filtering by booking state
CREATE INDEX idx_bookings_status ON bookings(status);
-- Index on confirmation_code for lookup
CREATE INDEX idx_bookings_confirmation ON bookings(confirmation_code);
-- Index on group_booking_id for finding sibling bookings
CREATE INDEX idx_bookings_group ON bookings(group_booking_id) WHERE group_booking_id IS NOT NULL;
-- Composite index for upcoming bookings (dashboard query)
CREATE INDEX idx_bookings_upcoming ON bookings(booked_by, status, created_at DESC)
```

```sql
  WHERE status IN ('pending', 'confirmed');
-- Index for AI suggestion tracking
CREATE INDEX idx_bookings_ai_suggestion ON bookings(ai_suggestion_id) WHERE ai_suggestion_id IS NOT NULL;

-- Add foreign key from credit_ledger to bookings
ALTER TABLE credit_ledger ADD CONSTRAINT fk_credit_ledger_booking
  FOREIGN KEY (booking_id) REFERENCES bookings(id) ON DELETE SET NULL;

COMMENT ON TABLE bookings IS 'Activity reservations with full lifecycle tracking and group booking support';



-- ============================================================================
-- WAITLIST TABLE
-- Queue for full activity sessions with automatic notification
-- Priority is first-come-first-served with 4-hour confirmation window
-- ============================================================================
CREATE TABLE waitlist (
  -- Primary identifier
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

  -- Reference to the activity session they're waiting for
  session_id UUID NOT NULL REFERENCES activity_sessions(id) ON DELETE CASCADE,

  -- Reference to the child wanting to attend
  child_id UUID NOT NULL REFERENCES children(id) ON DELETE CASCADE,

  -- Reference to the parent on the waitlist
  user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

  -- Position in the waitlist queue (auto-incremented per session)
  position INTEGER NOT NULL,

  -- Timestamp when joined waitlist (for FIFO ordering)
  joined_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

  -- Notification tracking when spot opens
  spot_offered_at TIMESTAMP WITH TIME ZONE,
  spot_expires_at TIMESTAMP WITH TIME ZONE,  -- 4 hours after offered

  -- Response tracking
  response VARCHAR(20) CHECK (response IN ('accepted', 'declined', 'expired')),
  responded_at TIMESTAMP WITH TIME ZONE,

  -- If accepted, reference to the created booking
  converted_booking_id UUID REFERENCES bookings(id),

  -- Active status (removed from waitlist or not)
```

```
    is_active BOOLEAN DEFAULT true,
    removed_at TIMESTAMP WITH TIME ZONE,
    removed_reason VARCHAR(50),  -- 'user_removed', 'spot_accepted', 'spot_declined', 'spot_expired'

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure a child can only be on waitlist once per session
    CONSTRAINT unique_waitlist_entry UNIQUE (session_id, child_id)
);

-- Index on session_id for finding waitlist for a session
CREATE INDEX idx_waitlist_session_id ON waitlist(session_id);
-- Index on user_id for parent's waitlist view
CREATE INDEX idx_waitlist_user_id ON waitlist(user_id);
-- Index on child_id for finding child's waitlist entries
CREATE INDEX idx_waitlist_child_id ON waitlist(child_id);
-- Index for finding active waitlist entries ordered by position
CREATE INDEX idx_waitlist_active ON waitlist(session_id, position) WHERE is_active = true;
-- Index for finding entries with pending spot offers (for expiration processing)
CREATE INDEX idx_waitlist_pending_offers ON waitlist(spot_expires_at)
    WHERE spot_offered_at IS NOT NULL AND response IS NULL;

COMMENT ON TABLE waitlist IS 'FIFO queue for full sessions with 4-hour confirmation window';
```

2.8 Feedback, Reviews, and Notifications Tables

```
sql
```

```sql
-- ==============================================================================
-- BOOKING FEEDBACK TABLE
-- Quick thumbs up/down feedback collected 1 hour post-activity
-- Used for real-time vector updates and venue scoring
-- ==============================================================================
CREATE TABLE booking_feedback (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the booking this feedback is for (one feedback per booking)
    booking_id UUID NOT NULL UNIQUE REFERENCES bookings(id) ON DELETE CASCADE,

    -- Simple sentiment: true = thumbs up, false = thumbs down
    is_positive BOOLEAN NOT NULL,

    -- Optional tags for positive feedback
    -- Structure: ["fun", "educational", "social", "creative"]
    positive_tags JSONB DEFAULT '[]'::jsonb,

    -- Optional tags for negative feedback
    -- Structure: ["organization", "content", "instructor", "facility"]
    negative_tags JSONB DEFAULT '[]'::jsonb,

    -- Optional text comment
    comment TEXT,

    -- Timestamp when feedback was submitted
    submitted_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Flag indicating if this feedback has been processed for vector update
    vector_processed BOOLEAN DEFAULT false,
    vector_processed_at TIMESTAMP WITH TIME ZONE
);

-- Index on booking_id (already has unique constraint)
-- Index for unprocessed feedback (for vector update batch job)
CREATE INDEX idx_feedback_unprocessed ON booking_feedback(vector_processed)
    WHERE vector_processed = false;

COMMENT ON TABLE booking_feedback IS 'Quick post-activity sentiment feedback for AI vector updates';


-- ==============================================================================
-- REVIEWS TABLE
-- Detailed reviews with star ratings for venue performance scoring
-- Aggregated for activity and venue average ratings
-- ==========================================================================
CREATE TABLE reviews (
  -- Primary identifier
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

  -- Reference to the booking this review is for (one review per booking)
  booking_id UUID NOT NULL UNIQUE REFERENCES bookings(id) ON DELETE CASCADE,

  -- Reference to the activity being reviewed
  activity_id UUID NOT NULL REFERENCES activities(id) ON DELETE CASCADE,

  -- Reference to the venue being reviewed
  venue_id UUID NOT NULL REFERENCES venues(id) ON DELETE CASCADE,

  -- Reference to the user who wrote the review
  user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

  -- Star rating (1-5)
  rating INTEGER NOT NULL CHECK (rating BETWEEN 1 AND 5),

  -- Would book again? (for repeat booking metric)
  would_book_again BOOLEAN,

  -- Would recommend venue? (for venue scoring)
  would_recommend_venue BOOLEAN,

  -- Written review text (optional)
  review_text TEXT,

  -- Review visibility (can be hidden by admin if inappropriate)
  is_visible BOOLEAN DEFAULT true,
  hidden_reason TEXT,
  hidden_by UUID REFERENCES users(id),
  hidden_at TIMESTAMP WITH TIME ZONE,

  -- Venue response to review (optional)
  venue_response TEXT,
  venue_responded_at TIMESTAMP WITH TIME ZONE,
  venue_responded_by UUID REFERENCES users(id),

  -- Moderation status
  is_flagged BOOLEAN DEFAULT false,
  flagged_reason TEXT,

  -- Timestamp when review was submitted
  submitted_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on activity_id for retrieving reviews for an activity
CREATE INDEX idx_reviews_activity_id ON reviews(activity_id);
-- Index on venue_id for venue performance scoring
CREATE INDEX idx_reviews_venue_id ON reviews(venue_id);
-- Index on user_id for user's review history
CREATE INDEX idx_reviews_user_id ON reviews(user_id);
-- Index on rating for aggregation queries
CREATE INDEX idx_reviews_rating ON reviews(rating);
-- Index on visible reviews for public display
CREATE INDEX idx_reviews_visible ON reviews(activity_id, is_visible, submitted_at DESC)
    WHERE is_visible = true;

COMMENT ON TABLE reviews IS 'Detailed reviews with ratings for venue performance scoring';



-- ============================================================================
-- NOTIFICATIONS TABLE
-- Multi-channel notification delivery tracking
-- Supports email, SMS, WhatsApp, push, and in-app notifications
-- ============================================================================
CREATE TABLE notifications (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the recipient user
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Notification type for categorization and filtering
    notification_type VARCHAR(50) NOT NULL,
    -- Types: 'booking_confirmed', 'booking_reminder', 'booking_cancelled_venue',
    --        'monthly_suggestions', 'waitlist_spot', 'feedback_request',
    --        'credits_expiring', 'new_activities', 'system_announcement'

    -- Notification priority
    priority notification_priority DEFAULT 'normal',

    -- Notification title/subject
    title VARCHAR(255) NOT NULL,

    -- Notification body content
    body TEXT NOT NULL,
    -- Structured data payload for rich notifications (e.g., booking details)
    data_payload JSONB DEFAULT '{}'::jsonb,

    -- Delivery channel used
    channel notification_channel NOT NULL,

    -- Deep link URL for in-app navigation
    action_url VARCHAR(500),

    -- Related entity references (polymorphic)
    related_entity_type VARCHAR(50),  -- 'booking', 'activity', 'waitlist', etc.
    related_entity_id UUID,

    -- Delivery status tracking
    is_sent BOOLEAN DEFAULT false,
    sent_at TIMESTAMP WITH TIME ZONE,
    send_error TEXT,

    -- External delivery ID (from email/SMS/push provider)
    external_delivery_id VARCHAR(255),

    -- Read/seen tracking
    is_read BOOLEAN DEFAULT false,
    read_at TIMESTAMP WITH TIME ZONE,

    -- Scheduled delivery (for future notifications like reminders)
    scheduled_for TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for retrieving user's notifications
CREATE INDEX idx_notifications_user_id ON notifications(user_id);
-- Index for unread notifications (notification center badge)
CREATE INDEX idx_notifications_unread ON notifications(user_id, is_read, created_at DESC)
    WHERE is_read = false;
-- Index on notification_type for filtering
CREATE INDEX idx_notifications_type ON notifications(notification_type);
-- Index on scheduled notifications for delivery processing
CREATE INDEX idx_notifications_scheduled ON notifications(scheduled_for)
    WHERE is_sent = false AND scheduled_for IS NOT NULL;
-- Index for pending sends
CREATE INDEX idx_notifications_pending ON notifications(is_sent, created_at) WHERE is_sent = false;
```

```sql
COMMENT ON TABLE notifications IS 'Multi-channel notification delivery with status tracking';
```

2.9 AI Curation and Suggestions Tables

```
sql
```

```sql
-- ===========================================================================
-- AI CURATION SUGGESTIONS TABLE
-- Monthly personalized activity suggestions generated by LangGraph agent
-- Supports individual and sibling group recommendations
-- NOTE: Match scores come from Pinecone vector similarity; this table stores
--       the final suggestions and tracks parent responses for learning
-- ===========================================================================
CREATE TABLE ai_suggestions (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the parent user receiving suggestions
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Billing period these suggestions are for
    suggestion_month DATE NOT NULL,  -- First day of the month

    -- Children these suggestions are for (can be single or multiple for siblings)
    -- Stored as JSONB array of child IDs
    child_ids JSONB NOT NULL,  -- Structure: ["uuid1", "uuid2"]

    -- Reference to the suggested activity session
    session_id UUID NOT NULL REFERENCES activity_sessions(id) ON DELETE CASCADE,

    -- Reference to the activity (denormalized for easier queries)
    activity_id UUID NOT NULL REFERENCES activities(id) ON DELETE CASCADE,

    -- Suggestion ranking (1, 2, 3 for top 3 suggestions)
    rank INTEGER CHECK (rank BETWEEN 1 AND 10),

    -- Is this a sibling/group suggestion?
    is_group_suggestion BOOLEAN DEFAULT false,

    -- AI explanation of why this was suggested (generated by LLM)
    explanation TEXT,

    -- =======================================================================
    -- PINECONE-DERIVED SCORES
    -- These scores come from Pinecone vector similarity search
    -- =======================================================================

    -- Overall match score from Pinecone (cosine similarity, 0-1)
    -- For single child: direct child→activity similarity
    -- For siblings: computed intersection score
    match_score DECIMAL(4, 3) CHECK (match_score BETWEEN 0 AND 1),
    -- Individual child fit scores for group suggestions (from Pinecone)
    -- Structure: { "child_uuid1": 0.85, "child_uuid2": 0.78 }
    individual_fit_scores JSONB,

    -- Pinecone query details for debugging/auditing
    pinecone_query_id VARCHAR(255),

    -- ==========================================================================
    -- PARENT RESPONSE TRACKING
    -- ==========================================================================

    -- Parent response to suggestion
    parent_response VARCHAR(20),  -- 'accepted', 'rejected', 'ignored'
    responded_at TIMESTAMP WITH TIME ZONE,

    -- If accepted, reference to the created booking
    converted_booking_id UUID REFERENCES bookings(id),

    -- Rejection reason if declined (used to improve future suggestions)
    rejection_reason TEXT,

    -- ==========================================================================
    -- AUDIT & DEBUGGING
    -- ==========================================================================

    -- LangSmith trace ID for AI decision auditing
    langsmith_trace_id VARCHAR(255),

    -- Generation metadata
    generated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Expiry (suggestions expire at end of month)
    expires_at TIMESTAMP WITH TIME ZONE,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for retrieving user's suggestions
CREATE INDEX idx_suggestions_user_id ON ai_suggestions(user_id);
-- Index on suggestion_month for current month queries
CREATE INDEX idx_suggestions_month ON ai_suggestions(suggestion_month);
-- Composite index for parent dashboard (current month suggestions)
CREATE INDEX idx_suggestions_current ON ai_suggestions(user_id, suggestion_month, rank)
    WHERE parent_response IS NULL;
-- Index on activity_id for suggestion analytics
```

```sql
CREATE INDEX idx_suggestions_activity_id ON ai_suggestions(activity_id);
-- GIN index on child_ids for finding suggestions for specific children
CREATE INDEX idx_suggestions_child_ids ON ai_suggestions USING GIN (child_ids);

COMMENT ON TABLE ai_suggestions IS 'Monthly AI-curated activity suggestions with sibling matching';



-- ============================================================================
-- NATURAL LANGUAGE SEARCH LOGS TABLE
-- Analytics and audit log for NL search queries
-- NOTE: Actual semantic search is performed in Pinecone; this table is for:
--   1. Tracking search accuracy (90% target per PRD Section 7.3)
--   2. Analyzing user search patterns for improvements
--   3. Debugging failed searches
--   4. Compliance auditing
-- ============================================================================
CREATE TABLE nl_search_logs (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the user who made the search
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Original natural language query text
    query_text TEXT NOT NULL,

    -- ========================================================================
    -- LLM PARSING RESULTS (for accuracy tracking)
    -- ========================================================================

    -- Parsed intent structure extracted by LLM before Pinecone query
    -- Structure: { "outdoor": true, "messy": false, "competitive": false,
    --              "age_appropriate_for": "child_uuid", "energy_level": "high" }
    parsed_intent JSONB NOT NULL,

    -- Confidence score of LLM parsing (0-1)
    parse_confidence DECIMAL(4, 3),

    -- Whether clarification was requested from user
    clarification_requested BOOLEAN DEFAULT false,
    clarification_question TEXT,
    clarification_response TEXT,

    -- ========================================================================
    -- PINECONE QUERY DETAILS
    -- ========================================================================
    -- Pinecone query ID for debugging/tracing
    pinecone_query_id VARCHAR(255),

    -- Filters applied to Pinecone query (derived from parsed_intent)
    -- Structure: { "category": {"$in": ["stem", "building"]}, "min_age": {"$lte": 7} }
    pinecone_filters JSONB,

    -- Number of results requested from Pinecone
    pinecone_top_k INTEGER DEFAULT 10,

    -- ============================================================================
    -- SEARCH RESULTS (IDs only - actual data in PostgreSQL)
    -- ============================================================================

    -- Activity IDs returned (ordered by relevance)
    result_activity_ids JSONB,  -- Structure: ["uuid1", "uuid2", ...]

    -- Pinecone similarity scores for each result
    result_scores JSONB,  -- Structure: [0.95, 0.88, 0.82, ...]

    -- Number of results returned
    result_count INTEGER,

    -- ============================================================================
    -- USER INTERACTION TRACKING (for search quality metrics)
    -- ============================================================================

    -- Did user interact with results?
    had_interaction BOOLEAN DEFAULT false,
    clicked_activity_id UUID REFERENCES activities(id),
    resulted_in_booking BOOLEAN DEFAULT false,

    -- User feedback on search quality (if provided)
    feedback_helpful BOOLEAN,
    feedback_comment TEXT,

    -- Did user fall back to structured filters?
    fallback_to_filters BOOLEAN DEFAULT false,

    -- ============================================================================
    -- PERFORMANCE & DEBUGGING
    -- ============================================================================

    -- LangSmith trace ID for AI decision auditing
    langsmith_trace_id VARCHAR(255),

    -- Response time breakdown (milliseconds)
```

```
    llm_parse_duration_ms INTEGER,      -- Time for LLM to parse intent
    pinecone_query_duration_ms INTEGER, -- Time for Pinecone vector search
    postgres_enrich_duration_ms INTEGER,-- Time to fetch full activity data from PG
    total_duration_ms INTEGER,          -- Total end-to-end time

    -- Error tracking
    had_error BOOLEAN DEFAULT false,
    error_type VARCHAR(50),  -- 'llm_parse_error', 'pinecone_timeout', 'no_results'
    error_message TEXT,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for user search history
CREATE INDEX idx_nl_search_user_id ON nl_search_logs(user_id);
-- Index on created_at for date range queries
CREATE INDEX idx_nl_search_created_at ON nl_search_logs(created_at);
-- GIN index on parsed_intent for analyzing common intents
CREATE INDEX idx_nl_search_intent ON nl_search_logs USING GIN (parsed_intent);
-- Index for analyzing conversion rates
CREATE INDEX idx_nl_search_conversion ON nl_search_logs(resulted_in_booking) WHERE resulted_in_booking = true;
-- Index for error analysis and debugging
CREATE INDEX idx_nl_search_errors ON nl_search_logs(had_error, error_type) WHERE had_error = true;
-- Index for search quality analysis (fallback rate)
CREATE INDEX idx_nl_search_fallback ON nl_search_logs(fallback_to_filters) WHERE fallback_to_filters = true;
-- Full-text search on query_text for pattern analysis
CREATE INDEX idx_nl_search_query ON nl_search_logs USING GIN (to_tsvector('english', query_text));

COMMENT ON TABLE nl_search_logs IS 'Analytics log for NL searches. Actual search performed in Pinecone.';



-- =========================================================================
-- PINECONE SYNC STATUS TABLE
-- Operational tracking for PostgreSQL ↔ Pinecone synchronization
-- Used by N8N workflows and monitoring dashboards
-- =========================================================================
CREATE TABLE pinecone_sync_status (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Entity type being synced
    entity_type VARCHAR(50) NOT NULL CHECK (entity_type IN ('activity', 'child', 'venue')),

    -- Reference to the entity (polymorphic)
    entity_id UUID NOT NULL,
    -- Pinecone namespace where vector is stored
    pinecone_namespace VARCHAR(100) NOT NULL,

    -- Pinecone vector ID
    pinecone_vector_id VARCHAR(255) NOT NULL,

    -- Sync status
    sync_status VARCHAR(20) DEFAULT 'pending'
        CHECK (sync_status IN ('pending', 'synced', 'failed', 'deleted')),

    -- Last successful sync
    last_synced_at TIMESTAMP WITH TIME ZONE,

    -- Last sync attempt (even if failed)
    last_attempt_at TIMESTAMP WITH TIME ZONE,

    -- Failure tracking
    failure_count INTEGER DEFAULT 0,
    last_error_message TEXT,

    -- Hash of source data (to detect changes without re-embedding)
    content_hash VARCHAR(64),

    -- Embedding model version
    embedding_model_version VARCHAR(50),

    -- Vector dimensions (for validation)
    vector_dimensions INTEGER,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure one sync record per entity
    CONSTRAINT unique_sync_entity UNIQUE (entity_type, entity_id)
);

-- Index for finding entities needing sync
CREATE INDEX idx_pinecone_sync_pending ON pinecone_sync_status(sync_status, entity_type)
    WHERE sync_status IN ('pending', 'failed');
-- Index for monitoring failed syncs
CREATE INDEX idx_pinecone_sync_failed ON pinecone_sync_status(failure_count)
    WHERE sync_status = 'failed';
-- Index on namespace for bulk operations
CREATE INDEX idx_pinecone_sync_namespace ON pinecone_sync_status(pinecone_namespace);
```

```sql
COMMENT ON TABLE pinecone_sync_status IS 'Tracks synchronization status between PostgreSQL and Pinecone vectors';
```

2.10 Audit and Compliance Tables

sql

```sql
-- ==========================================================================
-- AUDIT LOGS TABLE
-- Comprehensive audit trail for compliance (COPPA/GDPR-K)
-- Retains logs for 2 years per PRD Section 10
-- ==========================================================================
CREATE TABLE audit_logs (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- User who performed the action (NULL for system actions)
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,

    -- Action category
    action_category VARCHAR(50) NOT NULL,
    -- Categories: 'auth', 'profile', 'booking', 'child', 'venue', 'admin', 'ai', 'system'

    -- Specific action performed
    action_type VARCHAR(100) NOT NULL,
    -- Examples: 'login', 'logout', 'profile_update', 'child_create', 'booking_create',
    --           'booking_cancel', 'consent_given', 'data_export', 'data_delete'

    -- Target entity type (polymorphic)
    entity_type VARCHAR(50),
    entity_id UUID,

    -- Previous and new values for change tracking (sensitive data encrypted)
    old_values JSONB,
    new_values JSONB,

    -- Request metadata for security analysis
    ip_address INET,
    user_agent TEXT,

    -- Session ID for grouping related actions
    session_id VARCHAR(255),

    -- Geographic location (derived from IP if available)
    geo_country VARCHAR(100),
    geo_city VARCHAR(100),

    -- Request ID for correlating with application logs
    request_id VARCHAR(255),

    -- Severity level
    severity VARCHAR(20) DEFAULT 'info' CHECK (severity IN ('debug', 'info', 'warning', 'error', 'critical')),
    -- Additional context as flexible JSONB
    context JSONB DEFAULT '{}'::jsonb,

    -- Timestamp (immutable)
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for user activity queries
CREATE INDEX idx_audit_logs_user_id ON audit_logs(user_id);
-- Index on action_category for filtering
CREATE INDEX idx_audit_logs_category ON audit_logs(action_category);
-- Index on action_type for specific action queries
CREATE INDEX idx_audit_logs_action ON audit_logs(action_type);
-- Index on entity for finding all actions on an entity
CREATE INDEX idx_audit_logs_entity ON audit_logs(entity_type, entity_id);
-- Index on created_at for date range queries and retention
CREATE INDEX idx_audit_logs_created_at ON audit_logs(created_at);
-- Index on severity for error analysis
CREATE INDEX idx_audit_logs_severity ON audit_logs(severity) WHERE severity IN ('warning', 'error', 'critical');
-- Composite index for compliance queries (user activity over time)
CREATE INDEX idx_audit_logs_compliance ON audit_logs(user_id, created_at DESC);

COMMENT ON TABLE audit_logs IS 'Comprehensive audit trail for COPPA/GDPR-K compliance (2-year retention)';


-- ============================================================================
-- PARENTAL CONSENT RECORDS TABLE
-- Explicit tracking of COPPA-required parental consent for child data
-- ============================================================================
CREATE TABLE parental_consent_records (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the parent giving consent
    parent_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Reference to the child consent is for
    child_id UUID NOT NULL REFERENCES children(id) ON DELETE CASCADE,

    -- Type of consent
    consent_type VARCHAR(50) NOT NULL,
    -- Types: 'data_collection', 'data_processing', 'activity_booking', 'photo_sharing'

    -- Consent version (tracks consent form version)
    consent_version VARCHAR(20) NOT NULL,

    -- Was consent given?
    consent_given BOOLEAN NOT NULL,

    -- Consent method
    consent_method VARCHAR(50) NOT NULL,  -- 'checkbox', 'signature', 'email_verification'

    -- Full consent text that was agreed to
    consent_text TEXT NOT NULL,

    -- IP address at time of consent
    ip_address INET,

    -- User agent at time of consent
    user_agent TEXT,

    -- Withdrawal tracking
    withdrawn BOOLEAN DEFAULT false,
    withdrawn_at TIMESTAMP WITH TIME ZONE,
    withdrawal_reason TEXT,

    -- Timestamps
    consented_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure one consent record per type per child (latest version)
    CONSTRAINT unique_consent_record UNIQUE (child_id, consent_type, consent_version)
);

-- Index on parent_id for consent management
CREATE INDEX idx_consent_parent_id ON parental_consent_records(parent_id);
-- Index on child_id for compliance checks
CREATE INDEX idx_consent_child_id ON parental_consent_records(child_id);
-- Index for active consents
CREATE INDEX idx_consent_active ON parental_consent_records(child_id, consent_type)
    WHERE consent_given = true AND withdrawn = false;

COMMENT ON TABLE parental_consent_records IS 'COPPA verifiable parental consent tracking';
```

2.11 Platform Configuration Tables

```sql
```

```sql
-- ==========================================================================
-- PLATFORM SETTINGS TABLE
-- System-wide configuration parameters managed by platform admins
-- ==========================================================================
CREATE TABLE platform_settings (
    -- Setting key (unique identifier)
    key VARCHAR(100) PRIMARY KEY,

    -- Setting value (stored as JSONB for flexibility)
    value JSONB NOT NULL,

    -- Human-readable description
    description TEXT,

    -- Setting category for grouping
    category VARCHAR(50),

    -- Data type hint for UI
    data_type VARCHAR(20) CHECK (data_type IN ('string', 'number', 'boolean', 'json', 'array')),

    -- Is this setting publicly visible?
    is_public BOOLEAN DEFAULT false,

    -- Last modified tracking
    updated_by UUID REFERENCES users(id),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Seed essential platform settings
INSERT INTO platform_settings (key, value, description, category, data_type) VALUES
    ('platform_fee_default', '17.5', 'Default platform fee percentage for new venues', 'financial', 'number'),
    ('cancellation_window_hours', '48', 'Hours before activity for full refund eligibility', 'booking', 'number'),
    ('waitlist_confirmation_hours', '4', 'Hours given to confirm waitlist spot', 'booking', 'number'),
    ('credit_expiry_warning_days', '7', 'Days before month end to send expiry warning', 'credits', 'number'),
    ('venue_score_curation_threshold', '60', 'Minimum venue score for AI curation eligibility', 'curation', 'number'),
    ('max_suggestions_per_month', '3', 'Maximum AI suggestions per child per month', 'curation', 'number'),
    ('feedback_request_delay_hours', '1', 'Hours after activity to send feedback request', 'feedback', 'number'),
    ('data_retention_days', '730', 'Days to retain audit logs (2 years)', 'compliance', 'number'),
    ('child_deletion_delay_days', '30', 'Days after deletion request to purge child data', 'compliance', 'number');

COMMENT ON TABLE platform_settings IS 'System-wide configuration parameters';


-- ==========================================================================
-- FEATURE FLAGS TABLE
-- Runtime feature toggles for gradual rollouts and A/B testing (UI Spec line 803)
-- ===============================================================================
CREATE TABLE feature_flags (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Feature flag key (unique identifier used in code)
    flag_key VARCHAR(100) NOT NULL UNIQUE,

    -- Human-readable name and description
    name VARCHAR(255) NOT NULL,
    description TEXT,

    -- Feature category for grouping
    category VARCHAR(50),  -- 'ai', 'booking', 'payment', 'ui', 'integration'

    -- Global enable/disable
    is_enabled BOOLEAN DEFAULT false,

    -- Percentage rollout (0-100, for gradual rollouts)
    rollout_percentage INTEGER DEFAULT 0 CHECK (rollout_percentage BETWEEN 0 AND 100),

    -- Target specific user roles (NULL means all roles)
    -- Structure: ["parent", "hr_admin", "venue_admin"]
    target_roles JSONB,

    -- Target specific corporate accounts (for beta testing with specific companies)
    target_corporate_ids JSONB DEFAULT '[]'::jsonb,

    -- Target specific users (for internal testing)
    target_user_ids JSONB DEFAULT '[]'::jsonb,

    -- Environment targeting
    environments JSONB DEFAULT '["production", "staging", "development"]'::jsonb,

    -- Feature flag variants for A/B testing
    -- Structure: { "control": 50, "variant_a": 25, "variant_b": 25 }
    variants JSONB,

    -- Start and end dates for time-limited features
    starts_at TIMESTAMP WITH TIME ZONE,
    ends_at TIMESTAMP WITH TIME ZONE,

    -- Audit tracking
    created_by UUID REFERENCES users(id),
    updated_by UUID REFERENCES users(id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
```

```
);

-- Index on flag_key for fast lookups
CREATE INDEX idx_feature_flags_key ON feature_flags(flag_key);
-- Index on enabled flags for runtime queries
CREATE INDEX idx_feature_flags_enabled ON feature_flags(is_enabled) WHERE is_enabled = true;
-- Index on category for admin filtering
CREATE INDEX idx_feature_flags_category ON feature_flags(category);

COMMENT ON TABLE feature_flags IS 'Runtime feature toggles for gradual rollouts and A/B testing';


-- ==========================================================================
-- NOTIFICATION TEMPLATES TABLE
-- Configurable notification content templates (UI Spec line 802)
-- ==========================================================================
CREATE TABLE notification_templates (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Template identifier used in code
    template_key VARCHAR(100) NOT NULL,

    -- Notification channel this template is for
    channel notification_channel NOT NULL,

    -- Language code for internationalization
    language_code VARCHAR(10) DEFAULT 'en',

    -- Template name for admin reference
    name VARCHAR(255) NOT NULL,

    -- Email-specific fields
    email_subject VARCHAR(255),
    email_body_html TEXT,
    email_body_text TEXT,

    -- SMS/WhatsApp/Push body
    short_body TEXT,

    -- In-app notification
    in_app_title VARCHAR(255),
    in_app_body TEXT,

    -- Available merge variables for this template
    -- Structure: ["parent_name", "child_name", "activity_name", "booking_date"]
    available_variables JSONB DEFAULT '[]'::jsonb,
    -- Template category for grouping
    category VARCHAR(50),  -- 'booking', 'reminder', 'waitlist', 'credits', 'system'

    -- Is this the active version?
    is_active BOOLEAN DEFAULT true,

    -- Version tracking for template history
    version INTEGER DEFAULT 1,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Ensure unique template per key, channel, and language
    CONSTRAINT unique_template_key_channel_lang UNIQUE (template_key, channel, language_code)
);

-- Index on template_key for lookups
CREATE INDEX idx_notification_templates_key ON notification_templates(template_key);
-- Index on channel for filtering
CREATE INDEX idx_notification_templates_channel ON notification_templates(channel);
-- Index on active templates
CREATE INDEX idx_notification_templates_active ON notification_templates(is_active) WHERE is_active = true;

COMMENT ON TABLE notification_templates IS 'Configurable notification content templates with multi-language support';


-- ================================================================================
-- DATA EXPORT REQUESTS TABLE
-- GDPR/CCPA data export tracking (UI Spec line 414 - "Download my data")
-- ================================================================================
CREATE TABLE data_export_requests (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the user requesting export
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Type of export
    export_type VARCHAR(50) NOT NULL,  -- 'full_data', 'children_data', 'booking_history', 'personal_info'

    -- Export format
    format VARCHAR(20) DEFAULT 'json',  -- 'json', 'csv', 'pdf'

    -- Request status
    status VARCHAR(20) DEFAULT 'pending'
```

```
        CHECK (status IN ('pending', 'processing', 'completed', 'failed', 'expired')),

    -- Processing timestamps
    requested_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    started_at TIMESTAMP WITH TIME ZONE,
    completed_at TIMESTAMP WITH TIME ZONE,

    -- Generated file details
    file_url VARCHAR(500),
    file_size_bytes BIGINT,
    file_expires_at TIMESTAMP WITH TIME ZONE,  -- Auto-delete after X days

    -- Download tracking
    download_count INTEGER DEFAULT 0,
    last_downloaded_at TIMESTAMP WITH TIME ZONE,

    -- Error tracking
    error_message TEXT,

    -- Request metadata (IP, user agent for compliance)
    request_ip INET,
    request_user_agent TEXT,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for listing user's export requests
CREATE INDEX idx_data_exports_user_id ON data_export_requests(user_id);
-- Index on status for processing queue
CREATE INDEX idx_data_exports_status ON data_export_requests(status) WHERE status IN ('pending', 'processing');
-- Index on expiry for cleanup job
CREATE INDEX idx_data_exports_expiry ON data_export_requests(file_expires_at)
    WHERE status = 'completed' AND file_expires_at IS NOT NULL;

COMMENT ON TABLE data_export_requests IS 'GDPR/CCPA data export request tracking';


-- ===========================================================================
-- CHAT CONVERSATIONS TABLE
-- Stores NL search conversation sessions for context and history
-- Supports multi-turn conversations with the AI search interface
-- ===========================================================================
CREATE TABLE chat_conversations (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    -- Reference to the user
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Conversation title (auto-generated or user-set)
    title VARCHAR(255),

    -- Conversation type
    conversation_type VARCHAR(50) DEFAULT 'activity_search',  -- 'activity_search', 'support', 'booking_help'

    -- Child context (if searching for specific child)
    child_id UUID REFERENCES children(id) ON DELETE SET NULL,

    -- Is conversation still active?
    is_active BOOLEAN DEFAULT true,

    -- Timestamps
    started_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    last_message_at TIMESTAMP WITH TIME ZONE,
    ended_at TIMESTAMP WITH TIME ZONE,

    -- Conversation summary (AI-generated for quick reference)
    summary TEXT,

    -- Outcome tracking
    resulted_in_booking BOOLEAN DEFAULT false,
    booking_id UUID REFERENCES bookings(id) ON DELETE SET NULL,

    -- Audit timestamps
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on user_id for user's conversation history
CREATE INDEX idx_chat_conversations_user_id ON chat_conversations(user_id);
-- Index on active conversations
CREATE INDEX idx_chat_conversations_active ON chat_conversations(user_id, is_active, last_message_at DESC)
    WHERE is_active = true;
-- Index on child_id for child-specific conversations
CREATE INDEX idx_chat_conversations_child ON chat_conversations(child_id) WHERE child_id IS NOT NULL;

COMMENT ON TABLE chat_conversations IS 'NL search conversation sessions for multi-turn AI interactions';


-- =============================================================================
-- CHAT MESSAGES TABLE
-- Individual messages within a conversation
-- =============================================================================
```

```sql
CREATE TABLE chat_messages (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Reference to the conversation
    conversation_id UUID NOT NULL REFERENCES chat_conversations(id) ON DELETE CASCADE,

    -- Message role
    role VARCHAR(20) NOT NULL CHECK (role IN ('user', 'assistant', 'system')),

    -- Message content
    content TEXT NOT NULL,

    -- For assistant messages: parsed intent (if applicable)
    parsed_intent JSONB,

    -- Activity results shown (if search was performed)
    result_activity_ids JSONB,

    -- Quick action chips shown to user
    suggested_actions JSONB,

    -- User interaction with this message
    user_clicked_activity_id UUID REFERENCES activities(id),
    user_selected_action VARCHAR(100),

    -- LangSmith trace for AI messages
    langsmith_trace_id VARCHAR(255),

    -- Timing
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    -- Message ordering within conversation
    sequence_number INTEGER NOT NULL
);

-- Index on conversation_id for fetching messages
CREATE INDEX idx_chat_messages_conversation ON chat_messages(conversation_id, sequence_number);
-- Index on role for filtering
CREATE INDEX idx_chat_messages_role ON chat_messages(conversation_id, role);

COMMENT ON TABLE chat_messages IS 'Individual messages in NL search conversations';


-- ==========================================================================
-- API KEYS TABLE
-- API key management for platform admin (UI Spec line 804)
-- ============================================================================
CREATE TABLE api_keys (
    -- Primary identifier
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Human-readable name for the API key
    name VARCHAR(255) NOT NULL,

    -- Description of what this key is used for
    description TEXT,

    -- The actual API key (hashed, only shown once on creation)
    key_hash VARCHAR(255) NOT NULL,

    -- Key prefix for identification (first 8 chars, stored unhashed)
    key_prefix VARCHAR(10) NOT NULL,

    -- Key type/purpose
    key_type VARCHAR(50) NOT NULL,  -- 'integration', 'testing', 'service', 'webhook'

    -- Permissions/scopes granted to this key
    -- Structure: ["read:activities", "write:bookings", "admin:venues"]
    scopes JSONB DEFAULT '[]'::jsonb,

    -- Rate limiting configuration
    rate_limit_per_minute INTEGER DEFAULT 60,
    rate_limit_per_day INTEGER DEFAULT 10000,

    -- IP whitelist (NULL means no restriction)
    allowed_ips JSONB,  -- Structure: ["192.168.1.1", "10.0.0.0/8"]

    -- Expiration
    expires_at TIMESTAMP WITH TIME ZONE,

    -- Status
    is_active BOOLEAN DEFAULT true,
    revoked_at TIMESTAMP WITH TIME ZONE,
    revoked_by UUID REFERENCES users(id),
    revoke_reason TEXT,

    -- Usage tracking
    last_used_at TIMESTAMP WITH TIME ZONE,
    last_used_ip INET,
    total_requests BIGINT DEFAULT 0,

    -- Created by (platform admin)
    created_by UUID NOT NULL REFERENCES users(id),
      -- Audit timestamps
      created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
      updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on key_prefix for lookup (full hash comparison done after)
CREATE INDEX idx_api_keys_prefix ON api_keys(key_prefix) WHERE is_active = true;
-- Index on key_type for filtering
CREATE INDEX idx_api_keys_type ON api_keys(key_type);
-- Index on expiration for cleanup
CREATE INDEX idx_api_keys_expiry ON api_keys(expires_at) WHERE is_active = true AND expires_at IS NOT NULL;

COMMENT ON TABLE api_keys IS 'API key management for integrations and external access';


-- ============================================================================
-- INVOICES TABLE
-- Corporate billing invoices generated monthly
-- ============================================================================
CREATE TABLE invoices (
      -- Primary identifier
      id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

      -- Invoice number for display (sequential per corporate)
      invoice_number VARCHAR(50) NOT NULL UNIQUE,

      -- Reference to the corporate account
      corporate_id UUID NOT NULL REFERENCES corporates(id) ON DELETE RESTRICT,

      -- Reference to the subscription
      subscription_id UUID NOT NULL REFERENCES corporate_subscriptions(id) ON DELETE RESTRICT,

      -- Billing period
      billing_period_start DATE NOT NULL,
      billing_period_end DATE NOT NULL,

      -- Invoice amounts
      subtotal DECIMAL(12, 2) NOT NULL CHECK (subtotal >= 0),
      tax_amount DECIMAL(12, 2) DEFAULT 0 CHECK (tax_amount >= 0),
      total_amount DECIMAL(12, 2) NOT NULL CHECK (total_amount >= 0),

      -- Currency
      currency VARCHAR(3) DEFAULT 'USD',

      -- Invoice status
      status VARCHAR(20) DEFAULT 'draft' CHECK (status IN ('draft', 'sent', 'paid', 'overdue', 'cancelled')),
      -- Due date (net-30 by default)
      due_date DATE NOT NULL,

      -- Payment tracking
      paid_at TIMESTAMP WITH TIME ZONE,
      paid_amount DECIMAL(12, 2),
      payment_method VARCHAR(50),
      payment_reference VARCHAR(255),

      -- Stripe invoice ID
      stripe_invoice_id VARCHAR(255),

      -- Invoice PDF URL
      pdf_url VARCHAR(500),

      -- Line items stored as JSONB for flexibility
      line_items JSONB NOT NULL,

      -- Notes
      notes TEXT,

      -- Sent tracking
      sent_at TIMESTAMP WITH TIME ZONE,

      -- Audit timestamps
      created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
      updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Index on corporate_id for retrieving invoices
CREATE INDEX idx_invoices_corporate_id ON invoices(corporate_id);
-- Index on status for filtering
CREATE INDEX idx_invoices_status ON invoices(status);
-- Index on due_date for overdue processing
CREATE INDEX idx_invoices_due_date ON invoices(due_date) WHERE status = 'sent';

COMMENT ON TABLE invoices IS 'Corporate billing invoices with Stripe integration';
```

2.12 Update Trigger Function

```
sql
```

```sql
-- ================================================================================
-- AUTOMATIC UPDATED_AT TRIGGER FUNCTION
-- Automatically updates the updated_at timestamp on row modification
-- ================================================================================
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ language 'plpgsql';

-- Apply trigger to all tables with updated_at column
CREATE TRIGGER update_corporates_updated_at BEFORE UPDATE ON corporates
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_users_updated_at BEFORE UPDATE ON users
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_user_locations_updated_at BEFORE UPDATE ON user_locations
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_venues_updated_at BEFORE UPDATE ON venues
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_activities_updated_at BEFORE UPDATE ON activities
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_activity_sessions_updated_at BEFORE UPDATE ON activity_sessions
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_children_updated_at BEFORE UPDATE ON children
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_corporate_subscriptions_updated_at BEFORE UPDATE ON corporate_subscriptions
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_user_credit_balances_updated_at BEFORE UPDATE ON user_credit_balances
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_bookings_updated_at BEFORE UPDATE ON bookings
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_waitlist_updated_at BEFORE UPDATE ON waitlist
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
```

```sql
CREATE TRIGGER update_reviews_updated_at BEFORE UPDATE ON reviews
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_ai_suggestions_updated_at BEFORE UPDATE ON ai_suggestions
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_invoices_updated_at BEFORE UPDATE ON invoices
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_pinecone_sync_status_updated_at BEFORE UPDATE ON pinecone_sync_status
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_feature_flags_updated_at BEFORE UPDATE ON feature_flags
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_notification_templates_updated_at BEFORE UPDATE ON notification_templates
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_chat_conversations_updated_at BEFORE UPDATE ON chat_conversations
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_api_keys_updated_at BEFORE UPDATE ON api_keys
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_employee_invitations_updated_at BEFORE UPDATE ON employee_invitations
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_activity_recurring_schedules_updated_at BEFORE UPDATE ON activity_recurring_schedules
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_venue_payouts_updated_at BEFORE UPDATE ON venue_payouts
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_venue_integrations_updated_at BEFORE UPDATE ON venue_integrations
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();


-- ============================================================================
-- TRIGGER: Mark activities for Pinecone re-sync on content changes
-- ============================================================================
CREATE OR REPLACE FUNCTION mark_activity_for_pinecone_sync()
RETURNS TRIGGER AS $$
BEGIN
  -- Only mark for sync if searchable content changed
  IF (OLD.name IS DISTINCT FROM NEW.name OR
      OLD.short_description IS DISTINCT FROM NEW.short_description OR
      OLD.full_description IS DISTINCT FROM NEW.full_description OR
```

```
        OLD.category IS DISTINCT FROM NEW.category OR
        OLD.activity_tags IS DISTINCT FROM NEW.activity_tags) THEN
        NEW.pinecone_sync_required = true;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_activity_pinecone_sync BEFORE UPDATE ON activities
    FOR EACH ROW EXECUTE FUNCTION mark_activity_for_pinecone_sync();



-- ============================================================================
-- TRIGGER: Mark children for Pinecone re-sync on preference changes
-- ============================================================================
CREATE OR REPLACE FUNCTION mark_child_for_pinecone_sync()
RETURNS TRIGGER AS $$
BEGIN
    -- Only mark for sync if preference-related fields changed
    IF (OLD.energy_level IS DISTINCT FROM NEW.energy_level OR
        OLD.social_preference IS DISTINCT FROM NEW.social_preference OR
        OLD.interests IS DISTINCT FROM NEW.interests OR
        OLD.activities_to_avoid IS DISTINCT FROM NEW.activities_to_avoid) THEN
        NEW.pinecone_sync_required = true;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_child_pinecone_sync BEFORE UPDATE ON children
    FOR EACH ROW EXECUTE FUNCTION mark_child_for_pinecone_sync();
```

---

3. Performance Optimizations

3.1 Materialized View for Dashboard Queries

```
sql
```

```sql
-- ===========================================================================
-- MATERIALIZED VIEW: PARENT DASHBOARD SUMMARY
-- Pre-computed view for fast parent dashboard loading
-- Refresh hourly via N8N workflow
-- ===========================================================================
CREATE MATERIALIZED VIEW parent_dashboard_summary AS
SELECT
    u.id AS user_id,
    u.first_name,
    u.last_name,
    ucb.credits_available,
    ucb.credits_used,
    ucb.credits_allocated,
    ucb.billing_period_start,
    ucb.billing_period_end,
    -- Days remaining calculation
    GREATEST(0, ucb.billing_period_end - CURRENT_DATE) AS days_remaining,
    -- Count of children
    (SELECT COUNT(*) FROM children c WHERE c.parent_id = u.id AND c.is_active = true) AS child_count,
    -- Count of upcoming bookings
    (SELECT COUNT(*)
     FROM bookings b
     JOIN activity_sessions s ON b.session_id = s.id
     WHERE b.booked_by = u.id
       AND b.status IN ('pending', 'confirmed')
       AND s.session_date >= CURRENT_DATE) AS upcoming_booking_count,
    -- Count of pending suggestions
    (SELECT COUNT(*)
     FROM ai_suggestions ais
     WHERE ais.user_id = u.id
       AND ais.parent_response IS NULL
       AND ais.expires_at > CURRENT_TIMESTAMP) AS pending_suggestion_count,
    -- Count of unread notifications
    (SELECT COUNT(*)
     FROM notifications n
     WHERE n.user_id = u.id
       AND n.is_read = false) AS unread_notification_count
FROM users u
LEFT JOIN user_credit_balances ucb ON ucb.user_id = u.id
    AND ucb.billing_period_start <= CURRENT_DATE
    AND ucb.billing_period_end >= CURRENT_DATE
WHERE u.role = 'parent' AND u.is_active = true;

-- Index on the materialized view
CREATE UNIQUE INDEX idx_dashboard_summary_user_id ON parent_dashboard_summary(user_id);
```

```
-- Function to refresh the materialized view
CREATE OR REPLACE FUNCTION refresh_parent_dashboard_summary()
RETURNS void AS $$
BEGIN
    REFRESH MATERIALIZED VIEW CONCURRENTLY parent_dashboard_summary;
END;
$$ LANGUAGE plpgsql;

COMMENT ON MATERIALIZED VIEW parent_dashboard_summary IS 'Pre-computed parent dashboard data refreshed ho
```

3.2 Partial Indexes for Common Query Patterns

```sql
-- ========================================================================
-- ADDITIONAL PARTIAL INDEXES FOR PERFORMANCE
-- Optimized for the most common query patterns identified in UI Spec
-- ========================================================================

-- Fast lookup of active activities by category and age (Browse Activities screen)
CREATE INDEX idx_activities_browse ON activities(category, min_age, max_age, average_rating DESC)
    WHERE is_active = true AND is_published = true;

-- Fast lookup of available sessions in the next 30 days
CREATE INDEX idx_sessions_next_30_days ON activity_sessions(session_date, activity_id)
    WHERE session_date BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '30 days'
    AND is_cancelled = false;

-- Fast lookup of bookings needing reminder (24 hours before)
CREATE INDEX idx_bookings_reminder_24h ON bookings(id)
    WHERE status = 'confirmed' AND reminder_24h_sent = false;

-- Fast lookup of expired waitlist offers
CREATE INDEX idx_waitlist_expired ON waitlist(spot_expires_at)
    WHERE is_active = true AND spot_offered_at IS NOT NULL AND response IS NULL;
```

---

4. Database Maintenance

4.1 Data Retention Policies

```sql
```

```sql
-- ============================================================================
-- DATA RETENTION AND CLEANUP
-- Implements retention policies per PRD compliance requirements
-- ============================================================================

-- Function to purge old audit logs (retain 2 years)
CREATE OR REPLACE FUNCTION purge_old_audit_logs()
RETURNS INTEGER AS $$
DECLARE
    deleted_count INTEGER;
BEGIN
    DELETE FROM audit_logs
    WHERE created_at < CURRENT_TIMESTAMP - INTERVAL '2 years';
    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    RETURN deleted_count;
END;
$$ LANGUAGE plpgsql;

-- Function to purge deleted children data (30 days after request)
CREATE OR REPLACE FUNCTION purge_deleted_children()
RETURNS INTEGER AS $$
DECLARE
    deleted_count INTEGER;
BEGIN
    DELETE FROM children
    WHERE deletion_requested = true
    AND deletion_scheduled_for <= CURRENT_DATE;
    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    RETURN deleted_count;
END;
$$ LANGUAGE plpgsql;

-- Function to purge old NL search logs (retain 90 days)
CREATE OR REPLACE FUNCTION purge_old_search_logs()
RETURNS INTEGER AS $$
DECLARE
    deleted_count INTEGER;
BEGIN
    DELETE FROM nl_search_logs
    WHERE created_at < CURRENT_TIMESTAMP - INTERVAL '90 days';
    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    RETURN deleted_count;
END;
$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION purge_old_audit_logs IS 'Removes audit logs older than 2 years';
COMMENT ON FUNCTION purge_deleted_children IS 'Purges child data 30 days after deletion request';
COMMENT ON FUNCTION purge_old_search_logs IS 'Removes NL search logs older than 90 days';

-- Function to purge expired data export files (retain 7 days)
CREATE OR REPLACE FUNCTION purge_expired_data_exports()
RETURNS INTEGER AS $$
DECLARE
    deleted_count INTEGER;
BEGIN
    UPDATE data_export_requests
    SET status = 'expired', file_url = NULL
    WHERE status = 'completed'
    AND file_expires_at < CURRENT_TIMESTAMP;
    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    RETURN deleted_count;
END;
$$ LANGUAGE plpgsql;

-- Function to purge old chat conversations (retain 90 days)
CREATE OR REPLACE FUNCTION purge_old_chat_conversations()
RETURNS INTEGER AS $$
DECLARE
    deleted_count INTEGER;
BEGIN
    DELETE FROM chat_conversations
    WHERE is_active = false
    AND ended_at < CURRENT_TIMESTAMP - INTERVAL '90 days';
    GET DIAGNOSTICS deleted_count = ROW_COUNT;
    RETURN deleted_count;
END;
$$ LANGUAGE plpgsql;

-- Function to expire pending employee invitations
CREATE OR REPLACE FUNCTION expire_pending_invitations()
RETURNS INTEGER AS $$
DECLARE
    updated_count INTEGER;
BEGIN
    UPDATE employee_invitations
    SET status = 'expired'
    WHERE status = 'pending'
    AND expires_at < CURRENT_TIMESTAMP;
    GET DIAGNOSTICS updated_count = ROW_COUNT;
    RETURN updated_count;
END;
$$ LANGUAGE plpgsql;
```

```
COMMENT ON FUNCTION purge_expired_data_exports IS 'Marks expired data exports and removes file URLs';
COMMENT ON FUNCTION purge_old_chat_conversations IS 'Removes inactive chat conversations older than 90 days';
COMMENT ON FUNCTION expire_pending_invitations IS 'Expires pending employee invitations past their expiry date';
```

---

5. Summary Statistics

Metric	Count
Total Tables	32
Enum Types	13
Indexes	95+
Foreign Keys	45+
Triggers	27
Materialized Views	1
Utility Functions	9

Tables by Domain

Domain	Tables
Users & Auth	`users`, `user_locations`, `corporates`, `employee_invitations`
Venues	`venues`, `venue_performance_scores`, `venue_payouts`, `venue_integrations`
Activities	`activities`, `activity_sessions`, `activity_recurring_schedules`
Children	`children`, `child_favorites`
Subscriptions	`corporate_subscriptions`, `user_credit_balances`, `credit_ledger`
Bookings	`bookings`, `waitlist`
Feedback	`booking_feedback`, `reviews`
Notifications	`notifications`, `notification_templates`
AI/ML & Pinecone	`ai_suggestions`, `nl_search_logs`, `pinecone_sync_status`, `chat_conversations`, `chat_messages`

Domain	Tables
Compliance	audit_logs, parental_consent_records, data_export_requests
Platform	platform_settings, invoices, feature_flags, api_keys

6. JSONB Column Summary

The following columns use JSONB for flexible, schema-less data storage:

Table	Column	Purpose
`users`	`notification_preferences`	User notification channel preferences
`venues`	`business_hours`	Operating hours by day of week
`venues`	`gallery_images`	Array of image URLs
`venues`	`accessibility_features`	Array of accessibility tags
`activities`	`learning_outcomes`	Array of learning outcome strings
`activities`	`what_to_bring`	Array of items to bring
`activities`	`activity_tags`	Key-value activity characteristics (synced to Pinecone metadata)
`activities`	`accessibility_features`	Array of accessibility tags
`activity_recurring_schedules`	`days_of_week`	Array of weekday numbers for schedule
`children`	`interests`	Array of interest categories (used in Pinecone embedding)
`children`	`activities_to_avoid`	Array of activity types to exclude (Pinecone filter)
`children`	`accessibility_needs`	Array of accessibility requirements
`venue_performance_scores`	`improvement_suggestions`	AI-generated improvement tips
`venue_integrations`	`api_config`	Integration-specific API configuration
`ai_suggestions`	`child_ids`	Array of child UUIDs for group suggestions
`ai_suggestions`	`individual_fit_scores`	Per-child Pinecone similarity scores
`nl_search_logs`	`parsed_intent`	LLM-extracted intent attributes
`nl_search_logs`	`pinecone_filters`	Filters applied to Pinecone query
`nl_search_logs`	`result_activity_ids`	Array of result UUIDs from Pinecone
`nl_search_logs`	`result_scores`	Pinecone similarity scores per result
`chat_messages`	`parsed_intent`	LLM-extracted intent for message
`chat_messages`	`result_activity_ids`	Activity results shown in chat
`chat_messages`	`suggested_actions`	Quick action chips shown to user

Table	Column	Purpose
`notifications`	`data_payload`	Rich notification context
`notification_templates`	`available_variables`	Merge variables for template
`feature_flags`	`target_roles`	Roles targeted by flag
`feature_flags`	`target_corporate_ids`	Corporates targeted by flag
`feature_flags`	`target_user_ids`	Users targeted by flag
`feature_flags`	`variants`	A/B test variant percentages
`api_keys`	`scopes`	Permission scopes for API key
`api_keys`	`allowed_ips`	IP whitelist for API key
`audit_logs`	`old_values`, `new_values`, `context`	Change tracking data
`invoices`	`line_items`	Invoice line item details

---

7. Pinecone ↔ PostgreSQL Data Flow

7.1 Activity Embedding Flow

```
┌─────────────────┐   CREATE/UPDATE   ┌─────────────────┐
│  activities   │ ────────────────────▶ │ Trigger sets   │
│  (PostgreSQL) │               │ sync_required   │
└─────────────────┘                     │ = true        │
                  └─────────────────┘
                        │
                        ▼
┌─────────────────┐   N8N Workflow   ┌─────────────────┐
│  Pinecone     │ ◀──────────────────── │ Sync Job      │
│  activity-    │  1. Fetch pending  │ (every 5 min) │
│  embeddings   │  2. Generate embed └─────────────────┘
│  namespace    │  3. Upsert vector
└─────────────────┘  4. Update sync_required = false
```

7.2 Search Query Flow

```
┌─────────────────┐         ┌─────────────────┐
│  User Query   │         │ LLM (Claude)  │
│ "outdoor,   │ ──────────────────────▶│ Parse Intent │
│ not messy"  │               └──────────┬──────────┘
└───────────────────┘                    │
                    ▼
             ┌───────────────────┐
             │ parsed_intent:  │
             │ {outdoor: true, │
             │  messy: false}  │
             └─────────┬─────────┘
                    │
      ┌──────────────────────────────────────────┘
      │
      ▼
┌───────────────────┐   Vector + Filter  ┌──────────────────┐
│ Pinecone     │◀────────────────────│ Query with   │
│ Search      │         │ metadata   │
└─────────┬─────────┘                  │ filters     │
       │                 └──────────────────┘
       │ Returns: [activity_ids, scores]
       ▼
┌───────────────────┐   Fetch full data  ┌──────────────────┐
│ PostgreSQL   │◀───────────────────│ Enrich with  │
│ activities  │          │ venue, session│
└─────────┬─────────┘                  │ availability │
       │                 └──────────────────┘
       │
       ▼
┌───────────────────┐
│ nl_search_  │ Log for analytics
│ logs      │
└───────────────────┘
```

7.3 Child Preference Update Flow (Feedback Loop)

```
┌───────────────────┐   Post-Activity   ┌──────────────────┐
│ booking_    │ ─────────────────────▶│ Process     │
│ feedback    │ Thumbs up/down     │ Feedback    │
└───────────────────┘            └──────────┬──────────┘
                    │
                    ▼
┌───────────────────┐   Update vector   ┌──────────────────┐
│ Pinecone     │◀────────────────────│ Adjust child │
│ child-     │ Shift preference  │ embedding    │
│ preferences  │ toward/away from  │ weights     │
│ namespace    │ activity type   └──────────────────┘
└───────────────────┘
```

```
┌────────────────┐
│ Update child  │
│ feedback_     │
│ incorporated  │
│ _count        │
└────────────────┘
```

Schema designed for Nexus Family Pass v7.0 - January 2026