```sql
-- ============================================================================
-- ClimateSage: Comprehensive SQL Schema
-- Version: 3.3 (Self-Contained Final Version)
-- Date: June 25, 2025
-- Description: This schema is the complete technical blueprint for the
--           application database. It is designed for PostgreSQL and
--           supports all defined user journeys, including complex,
--           multi-site, multi-framework scenarios.
-- ============================================================================


-- ============================================
-- PREAMBLE: ENABLE EXTENSIONS
-- ============================================
-- This schema uses UUIDs for primary keys to ensure global uniqueness
-- and better performance in distributed environments. We must first
-- enable the "uuid-ossp" extension in PostgreSQL.

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";


-- ============================================
-- SECTION 1: CORE ORGANIZATION, SITE, & USER TABLES
-- ============================================
-- These tables define the fundamental entities of the platform: our
-- client organizations, their physical locations (sites), and the
-- user profiles for individuals who access the system.


-- Table: organizations
-- Stores master information about each client organization.
CREATE TABLE organizations (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  name VARCHAR(255) NOT NULL,
  sector VARCHAR(100),
  industry_description TEXT,
  country_code VARCHAR(3),
  logo_url TEXT,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
COMMENT ON TABLE organizations IS 'Stores master information about each client organization.';
COMMENT ON COLUMN organizations.industry_description IS 'User-provided details about their specific niche (e.g., "5-star luxury hotel", "Plastics manufacturing").';



-- Table: sites
-- Stores information about each specific operational site or facility.
-- Now supports a parent-child hierarchy for sub-locations.
CREATE TABLE sites (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  organization_id UUID NOT NULL,
  parent_site_id UUID NULL, -- Enables sub-locations. NULL indicates a primary site.
  name VARCHAR(255) NOT NULL,
   address TEXT,
   city VARCHAR(100),
   country_code VARCHAR(3),
   created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
   CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
   CONSTRAINT fk_parent_site FOREIGN KEY(parent_site_id) REFERENCES sites(id) ON DELETE CASCADE -- Self-referencing key for hierarchy.
);
COMMENT ON TABLE sites IS 'Stores information about each specific operational site or facility, with support for sub-locations.';
COMMENT ON COLUMN sites.parent_site_id IS 'If NULL, this is a primary site. If populated, it links to a parent site, making this a sub-location.';


-- Table: profiles
-- Extends the authentication system's user table (e.g., Supabase auth.users)
-- to store app-specific data about a user.
CREATE TABLE profiles (
   id UUID PRIMARY KEY, -- This ID MUST match the ID in the auth.users table.
   full_name VARCHAR(255),
   avatar_url TEXT,
   created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
COMMENT ON TABLE profiles IS 'Stores app-specific data for each user. Extends the authentication table.';
```

```
-- ===============================================
-- SECTION 2: TEAM, ROLES, & FRAMEWORK MANAGEMENT
-- ===============================================
-- This section defines the relationships between users, organizations,
-- and sites, and manages which compliance frameworks are active for
-- each organization. This is the heart of the RBAC and personalization system.


-- Table: user_permissions
-- A crucial "join table" that links users to organizations and assigns them
-- a specific role with a defined scope (organization-wide or site-specific).
CREATE TABLE user_permissions (
   user_id UUID NOT NULL,
   organization_id UUID NOT NULL,
   site_id UUID NULL,
   role VARCHAR(50) NOT NULL, -- e.g., 'org_admin', 'site_admin', 'contributor', 'viewer'
   created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
   PRIMARY KEY (user_id, organization_id, site_id),
   CONSTRAINT fk_user FOREIGN KEY(user_id) REFERENCES profiles(id) ON DELETE CASCADE,
   CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
   CONSTRAINT fk_site FOREIGN KEY(site_id) REFERENCES sites(id) ON DELETE CASCADE
);
COMMENT ON TABLE user_permissions IS 'Links users to organizations/sites and assigns roles. The core of our RBAC system.';
```

```sql
COMMENT ON COLUMN user_permissions.site_id IS 'If NULL, role applies to the entire organization. If specified, role is limited to this site.';

COMMENT ON COLUMN user_permissions.role IS 'Role assigned to the user. E.g., ''org_admin'', ''site_admin'', ''contributor'', ''viewer''.';



-- Table: invitations
-- Manages the lifecycle of inviting new users to an organization.
CREATE TABLE invitations (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   organization_id UUID NOT NULL,
   invited_by_user_id UUID NOT NULL,
   email VARCHAR(255) NOT NULL,
   role VARCHAR(50) NOT NULL, -- The role the user will be assigned.
   site_id UUID NULL,
   token VARCHAR(255) NOT NULL UNIQUE,
   status VARCHAR(50) NOT NULL DEFAULT 'pending', -- 'pending', 'accepted', 'revoked', 'expired'
   expires_at TIMESTAMPTZ NOT NULL,
   created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
   CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
   CONSTRAINT fk_invited_by FOREIGN KEY(invited_by_user_id) REFERENCES profiles(id) ON DELETE CASCADE,
   CONSTRAINT fk_site FOREIGN KEY(site_id) REFERENCES sites(id) ON DELETE CASCADE
);
COMMENT ON TABLE invitations IS 'Manages the lifecycle of inviting new users to an organization.';
```

```
-- Table: frameworks
-- A master list of all supported compliance frameworks.
CREATE TABLE frameworks (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   framework_name VARCHAR(255) NOT NULL UNIQUE,
   description TEXT
);
COMMENT ON TABLE frameworks IS 'A master list of all supported compliance frameworks (e.g., ESG, DST, Green Key).';


-- Table: organization_frameworks
-- Links organizations to the frameworks they must comply with, acting as a switchboard.
CREATE TABLE organization_frameworks (
   organization_id UUID NOT NULL,
   framework_id UUID NOT NULL,
   is_active BOOLEAN NOT NULL DEFAULT true,
   activated_at TIMESTAMPTZ NOT NULL DEFAULT now(),
   PRIMARY KEY (organization_id, framework_id),
   CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
   CONSTRAINT fk_framework FOREIGN KEY(framework_id) REFERENCES frameworks(id) ON DELETE CASCADE
);
```

```
COMMENT ON TABLE organization_frameworks IS 'Acts as a switchboard to activate specific compliance frameworks for an organization.';



-- ========================================
-- SECTION 3: WIZARD, DATA & EVIDENCE TABLES
-- ========================================
-- These tables form the intelligent data collection engine. This includes the
-- master question library, the tagging system that drives personalization,
-- and the tables for storing answers and evidence.

-- Table: questions
-- The master library of all questions that can be asked in the Data Wizard.
CREATE TABLE questions (
  id VARCHAR(100) PRIMARY KEY, -- This is the 'metric_id', e.g., 'electricity_consumption_kwh'
  question_text TEXT NOT NULL,
  question_type VARCHAR(50) NOT NULL, -- e.g., 'numeric', 'text', 'boolean'
  helper_text TEXT
);
COMMENT ON TABLE questions IS 'The master library of all questions that can be asked in the Data Wizard.';
COMMENT ON COLUMN questions.id IS 'The unique metric_id for this question, used for programmatic reference.';



-- Table: question_tags
-- The tagging system that powers the intelligent wizard by linking questions
-- to frameworks, industries, topics, etc.
CREATE TABLE question_tags (
   question_id VARCHAR(100) NOT NULL,
   tag_type VARCHAR(50) NOT NULL, -- e.g., 'framework', 'industry', 'topic'
   tag_value VARCHAR(100) NOT NULL,
   PRIMARY KEY (question_id, tag_type, tag_value),
   CONSTRAINT fk_question FOREIGN KEY(question_id) REFERENCES questions(id) ON DELETE CASCADE
);
COMMENT ON TABLE question_tags IS 'The tagging system that powers the intelligent wizard by linking questions to frameworks and industries.';


-- Table: data_records
-- The central table for all quantitative and qualitative data points.
-- Its structure allows us to support any framework seamlessly.
CREATE TABLE data_records (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   organization_id UUID NOT NULL,
   site_id UUID NOT NULL,
   entered_by_user_id UUID NOT NULL,
   metric_id VARCHAR(100) NOT NULL, -- This now directly relates to the 'questions.id'
   value_numeric DECIMAL(18, 4),
   value_text TEXT,
   value_boolean BOOLEAN,
   period_start DATE NOT NULL,
```

```sql
  period_end DATE NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
  CONSTRAINT fk_site FOREIGN KEY(site_id) REFERENCES sites(id) ON DELETE CASCADE,
  CONSTRAINT fk_user FOREIGN KEY(entered_by_user_id) REFERENCES profiles(id) ON DELETE SET NULL,
  CONSTRAINT fk_question_metric FOREIGN KEY(metric_id) REFERENCES questions(id) ON DELETE RESTRICT
);
COMMENT ON TABLE data_records IS 'The core data table where all ESG, DST, and Green Key metrics are stored.';
COMMENT ON COLUMN data_records.metric_id IS 'The unique ID of the question being answered. Links to the questions table.';


-- Table: evidence_files
-- Stores metadata for all uploaded supporting documents and links them
-- directly to the data record they prove.
CREATE TABLE evidence_files (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  data_record_id UUID NOT NULL, -- Links evidence directly to a piece of data.
  organization_id UUID NOT NULL,
  uploaded_by_user_id UUID NOT NULL,
  file_name VARCHAR(255) NOT NULL,
  storage_path TEXT NOT NULL UNIQUE,
  file_hash VARCHAR(64) NOT NULL, -- SHA-256 hash to detect duplicates.
```

```sql
    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
    CONSTRAINT fk_data_record FOREIGN KEY(data_record_id) REFERENCES data_records(id) ON DELETE CASCADE,
    CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
    CONSTRAINT fk_user FOREIGN KEY(uploaded_by_user_id) REFERENCES profiles(id) ON DELETE SET NULL
);
COMMENT ON TABLE evidence_files IS 'Stores metadata for all uploaded supporting evidence files.';
COMMENT ON COLUMN evidence_files.data_record_id IS 'Direct link to the specific data_record this file proves, creating a clear audit trail.';



-- ================================================
-- SECTION 4: GOVERNANCE & AUDITING TABLES
-- ================================================
-- This section is essential for providing enterprise-grade traceability,
-- accountability, and a complete, immutable audit trail.

-- Table: audit_log
-- Logs every important change to data for a complete, immutable audit trail.
CREATE TABLE audit_log (
    id BIGINT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    organization_id UUID NOT NULL,
    user_id UUID, -- Can be NULL for system-level actions
    action VARCHAR(50) NOT NULL, -- e.g., 'CREATE', 'UPDATE', 'DELETE', 'LOGIN', 'INVITE'
```

```
   table_name VARCHAR(100),
   record_id UUID,
   diff_json JSONB, -- An object containing old and new values for updates.
   created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
   CONSTRAINT fk_organization FOREIGN KEY(organization_id) REFERENCES organizations(id) ON DELETE CASCADE,
   CONSTRAINT fk_user FOREIGN KEY(user_id) REFERENCES profiles(id) ON DELETE SET NULL
);
COMMENT ON TABLE audit_log IS 'Logs every important change to data for a complete, immutable audit trail.';
COMMENT ON COLUMN audit_log.diff_json IS 'For UPDATE actions, stores a JSON object showing { "old": {...}, "new": {...}}.';


-- ============================================
-- END OF SCHEMA v3.3
-- ============================================
```