```
"""
Master Pharmacognosy Database for Cannabis sativa
Comprehensive chemical compound profiles with grow-style dependencies,
temperature thresholds, therapeutic benefits, and adverse reactions
"""

import sqlite3
from typing import List, Dict, Optional, Tuple
from dataclasses import dataclass
from enum import Enum
from datetime import datetime


# ============================================================================
# ENUMERATIONS - Input Parameters
# ============================================================================

class GrowStyle(Enum):
    """Cultivation vectors determining baseline chemical potential"""
    LIVING_SOIL = "living_soil"       # High diversity, sesquiterpene boost
    HYDROPONIC = "hydroponic"         # High potency, low diversity
    SUN_GROWN = "sun_grown"           # UV defense, flavonoid boost
    DROUGHT_STRESS = "drought_stress" # Metabolic concentration

class InterfaceTemp(Enum):
    """Temperature ranges determining compound bioavailability"""
    LOW_TEMP = "low"      # <180°C - Volatile terpenes only
    MED_TEMP = "medium"   # 180-220°C - Optimal cannabinoid/cannflavin
    HIGH_TEMP = "high"    # 230-600°C - Quercetin activation, benzene risk
    COMBUSTION = "combustion"  # >600°C - Full spectrum + toxicants


# ============================================================================
# DATA MODELS
# ============================================================================

@dataclass
class TerpeneProfile:
    """Terpene compound with thermodynamic and therapeutic properties"""
    id: str
    name: str
    boiling_point: float  # °C
    primary_benefit: str
    adverse_reaction: str
    grow_dependency: str
    logic_role: str  # "Modulator" - dictates direction of effects

@dataclass
class CannabinoidProfile:
    """Cannabinoid compound with receptor targets and biphasic properties"""
```

```
    id: str
    name: str
    boiling_point: float  # °C
    primary_benefit: str
    adverse_reaction: str
    logic_flag: str  # "Driver" - dictates intensity
    receptor_target: str  # CB1, CB2, etc.

@dataclass
class FlavonoidProfile:
    """Flavonoid compound with interface and cultivation requirements"""
    id: str
    name: str
    boiling_point: Optional[float]  # °C, None if requires combustion
    primary_benefit: str
    adverse_reaction: str
    grow_requirement: str
    interface_requirement: str
    potency_multiplier: float  # vs baseline compounds

@dataclass
class GrowStyleModifier:
    """Multipliers applied based on cultivation method"""
    grow_style: GrowStyle
    affected_compounds: List[str]
    multiplier: float
    diversity_flag: bool
    description: str


# ============================================================================
# PHARMACOGNOSY DATABASE MANAGER
# ============================================================================

class PharmacognosyDatabase:
    """Master database for cannabis chemical profiles and interactions"""

    def __init__(self, db_path: str = "pharmacognosy.db"):
        self.db_path = db_path
        self.conn = sqlite3.connect(db_path)
        self._initialize_schema()
        self._populate_master_data()

    def _initialize_schema(self):
        """Create comprehensive schema for all chemical classes"""
        cursor = self.conn.cursor()

        # Terpene profiles
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS terpenes (
```

```
        id TEXT PRIMARY KEY,
        name TEXT NOT NULL,
        boiling_point REAL NOT NULL,
        primary_benefit TEXT NOT NULL,
        adverse_reaction TEXT NOT NULL,
        grow_dependency TEXT NOT NULL,
        logic_role TEXT DEFAULT 'modulator',
        citation_refs TEXT,
        last_updated TIMESTAMP
    )
""")

# Cannabinoid profiles
cursor.execute("""
    CREATE TABLE IF NOT EXISTS cannabinoids (
        id TEXT PRIMARY KEY,
        name TEXT NOT NULL,
        boiling_point REAL NOT NULL,
        primary_benefit TEXT NOT NULL,
        adverse_reaction TEXT NOT NULL,
        logic_flag TEXT NOT NULL,
        receptor_target TEXT,
        psychoactive BOOLEAN DEFAULT 0,
        citation_refs TEXT,
        last_updated TIMESTAMP
    )
""")

# Flavonoid profiles
cursor.execute("""
    CREATE TABLE IF NOT EXISTS flavonoids (
        id TEXT PRIMARY KEY,
        name TEXT NOT NULL,
        boiling_point REAL,
        primary_benefit TEXT NOT NULL,
        adverse_reaction TEXT NOT NULL,
        grow_requirement TEXT NOT NULL,
        interface_requirement TEXT NOT NULL,
        potency_multiplier REAL DEFAULT 1.0,
        citation_refs TEXT,
        last_updated TIMESTAMP
    )
""")

# Grow style modifiers
cursor.execute("""
    CREATE TABLE IF NOT EXISTS grow_modifiers (
        grow_style TEXT,
        compound_class TEXT,
        compound_names TEXT,
```

```
            multiplier REAL NOT NULL,
            diversity_flag BOOLEAN,
            description TEXT,
            PRIMARY KEY (grow_style, compound_class)
        )
    """)

    # Synergy rules (entourage effects)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS synergy_rules (
            rule_id TEXT PRIMARY KEY,
            rule_name TEXT NOT NULL,
            condition TEXT NOT NULL,
            effect TEXT NOT NULL,
            multiplier REAL NOT NULL,
            citation_refs TEXT
        )
    """)

    # Safety warnings
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS safety_warnings (
            warning_id TEXT PRIMARY KEY,
            trigger_condition TEXT NOT NULL,
            warning_level TEXT NOT NULL,
            message TEXT NOT NULL,
            compounds_involved TEXT,
            citation_refs TEXT
        )
    """)

    # Temperature profiles
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS temp_profiles (
            temp_range_name TEXT PRIMARY KEY,
            temp_min REAL NOT NULL,
            temp_max REAL,
            description TEXT NOT NULL,
            active_compounds TEXT,
            risk_level TEXT,
            recommended_for TEXT
        )
    """)

    self.conn.commit()

def _populate_master_data(self):
    """Load all pharmacognosy data"""
    cursor = self.conn.cursor()
```

```
# Check if already populated
cursor.execute("SELECT COUNT(*) FROM terpenes")
if cursor.fetchone()[0] > 0:
    return

# ========== TERPENE DATA ==========
terpenes = [
    ("T-01", "Myrcene", 168,
     "Sedation/Analgesia (Muscle Relaxant)",
     "Couch-Lock (Lethargy), Drowsiness",
     "Stress Response: Upregulated by abiotic stress/drought",
     "[7,11,12]"),

    ("T-02", "Limonene", 176,
     "Anxiolytic (Anti-Anxiety), Mood Elevation",
     "Anxiety (Paradoxical at high dose), Skin Irritant",
     "General: Stable across grow styles",
     "[13,14,15]"),

    ("T-03", "Pinene", 155,
     "Memory Retention, Bronchodilation",
     "Racing Thoughts, Anxiety in sensitive users",
     "Lighting: Synthesis influenced by light intensity/spectrum",
     "[5,15,16]"),

    ("T-04", "Linalool", 198,
     "Sedation/Calm (GABA modulation)",
     "Sedation (Excessive), potential allergen",
     "Terroir: Often higher in diverse soil microbiomes",
     "[14,16]"),

    ("T-05", "β-Caryophyllene", 263,
     "Pain/Inflammation (CB2 Agonist)",
     "Mild Irritation (Respiratory/Skin)",
     "Living Soil: Significantly higher in outdoor/soil grows",
     "[1,17,18]"),

    ("T-06", "Humulene", 276,
     "Appetite Suppression, Anti-inflammatory",
     "Anorectic (Unwanted weight loss)",
     "Living Soil: Higher levels in outdoor/soil grows",
     "[1,19]"),

    ("T-07", "Terpinolene", 185,
     "Antioxidant, Sedative (in isolation)",
     "Anxiety/Jitters (Common in Sativa strains)",
     "Genetics: Highly dependent on cultivar genetics",
     "[15,20]"),

    ("T-08", "Ocimene", 100,
```

```
    "Decongestant, Antiviral",
    "Coughing (Severe lung irritation/cough reflex)",
    "Fragile: Easily lost in high-heat or poor curing",
    "[19,21]")
]

for terp in terpenes:
    cursor.execute("""
        INSERT INTO terpenes
        (id, name, boiling_point, primary_benefit, adverse_reaction,
         grow_dependency, logic_role, citation_refs, last_updated)
        VALUES (?, ?, ?, ?, ?, ?, 'modulator', ?, ?)
    """, (*terp, datetime.now()))

# ========== CANNABINOID DATA ==========
cannabinoids = [
    ("C-01", "THC (Δ⁹)", 157,
     "Analgesia, Euphoria, Antiemetic",
     "Anxiety/Paranoia, Tachycardia, Memory Loss",
     "Psychoactive Engine", "CB1, CB2", 1, "[22,23]"),

    ("C-02", "CBD", 180,
     "Anti-Seizure, Anxiolytic, Anti-inflammatory",
     "Liver Toxicity (High dose), Drug Interactions (CYP450)",
     "The Buffer (Reduces THC panic)", "CB1 (antagonist), CB2", 0, "[24,25]"),

    ("C-03", "CBG", 105,
     "Gut Health (IBD), Neuroprotection",
     "Digestive Upset, Appetite Stimulation",
     "The Mother (Precursor)", "CB1, CB2", 0, "[26,27]"),

    ("C-04", "CBN", 185,
     "Sedation (Sleep Aid)",
     "Grogginess (Next day), Drowsiness",
     "The Degrader (Aged/oxidized biomass)", "CB1 (weak)", 0, "[28]"),

    ("C-05", "THCV", 220,
     "Appetite Suppression, Energy, Diabetes",
     "Nausea, Anxiety (High dose only)",
     "Diet Weed", "CB1 (antagonist), CB2", 0, "[26,29]"),

    ("C-06", "CBC", 220,
     "Antidepressant, Acne reduction",
     "Dizziness, Fatigue",
     "Potentiator (Boosts other cannabinoids)", "Non-cannabinoid receptors", 0, "[30,31]")
]

for canna in cannabinoids:
    cursor.execute("""
        INSERT INTO cannabinoids
```

```
        (id, name, boiling_point, primary_benefit, adverse_reaction,
         logic_flag, receptor_target, psychoactive, citation_refs, last_updated)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    """, (*canna, datetime.now()))

# ========== FLAVONOID DATA ==========
flavonoids = [
    ("F-01", "Cannflavin A", 182,
     "Pain (30x Aspirin potency)",
     "Neurotoxicity (Only at extremely high in vitro doses)",
     "Any (Genetic), UV possibly enhances",
     "Med-High Vape (185°C+)",
     30.0, "[32,33,34]"),

    ("F-02", "Cannflavin B", 182,
     "Anti-inflammatory (5-LOX inhibitor)",
     "Unknown (Lack of human toxicity data)",
     "Any (Genetic)",
     "Med-High Vape (185°C+)",
     15.0, "[32,35]"),

    ("F-03", "Quercetin", 250,
     "Antiviral, Antioxidant",
     "Drug Interaction (Blood thinners), Headache",
     "UV-B Radiation (Sunscreen response)",
     "Combustion or Max Vape (>230°C)",
     5.0, "[5,36,37]"),

    ("F-04", "Apigenin", 178,
     "Anxiolytic (Calming/GABA)",
     "Sedation, CYP2C9 inhibition (Drug interaction)",
     "Nutrient Balance",
     "Low-Med Vape (178°C+)",
     3.0, "[38,39]"),

    ("F-05", "Kaempferol", None,
     "Anti-cancer, Osteogenic",
     "Iron Malabsorption (if oral), Drug interaction",
     "UV Stress",
     "Combustion (High thermal requirement)",
     4.0, "[40,41]")
]

for flav in flavonoids:
    cursor.execute("""
        INSERT INTO flavonoids
        (id, name, boiling_point, primary_benefit, adverse_reaction,
         grow_requirement, interface_requirement, potency_multiplier,
         citation_refs, last_updated)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

```
    """, (*flav, datetime.now()))

# ========== GROW STYLE MODIFIERS ==========
grow_mods = [
    ("living_soil", "sesquiterpenes", "β-Caryophyllene,Humulene", 1.25, 1,
     "High diversity flag: 25% boost to sesquiterpenes and minor flavonoids"),

    ("living_soil", "flavonoids", "minor_flavonoids", 1.25, 1,
     "Enhanced soil microbiome increases flavonoid diversity"),

    ("hydroponic", "cannabinoids", "THC,CBD", 1.10, 0,
     "High potency flag: 10% boost to major cannabinoids"),

    ("hydroponic", "flavonoids", "all", 0.80, 0,
     "Low diversity penalty: 20% reduction in flavonoid expression"),

    ("sun_grown", "flavonoids", "Quercetin,Cannflavin A,Cannflavin B", 1.40, 1,
     "UV defense flag: 40% boost to UV-responsive flavonoids and anthocyanins"),

    ("drought_stress", "all", "cannabinoids,terpenes", 1.15, 0,
     "Metabolic concentration: Increased density per gram, reduced yield")
]

for mod in grow_mods:
    cursor.execute("""
        INSERT INTO grow_modifiers
        (grow_style, compound_class, compound_names, multiplier,
         diversity_flag, description)
        VALUES (?, ?, ?, ?, ?, ?)
    """, mod)

# ========== SYNERGY RULES (Entourage Effects) ==========
synergy_rules = [
    ("SYN-01", "Anxiety Reduction",
     "Limonene > threshold OR CBD > threshold",
     "Anxiety score multiplied",
     0.6, "[13]"),

    ("SYN-02", "Sedation Enhancement",
     "Myrcene > 0.5% AND THC > 15%",
     "Sedation score multiplied",
     2.0, "[12]"),

    ("SYN-03", "Pain Relief (Flavonoid Boost)",
     "Cannflavin A present AND temp >= 182°C",
     "Pain relief score multiplied",
     1.5, "[33]"),

    ("SYN-04", "Memory Protection",
     "Pinene present AND CBD > 5%",
```

```
     "Reduces THC memory impairment",
     0.7, "[16,24]"),

    ("SYN-05", "Anti-inflammatory Cascade",
     "β-Caryophyllene present AND CBD present",
     "CB2 receptor activation enhanced",
     1.3, "[17,24]")
]

for rule in synergy_rules:
    cursor.execute("""
        INSERT INTO synergy_rules
        (rule_id, rule_name, condition, effect, multiplier, citation_refs)
        VALUES (?, ?, ?, ?, ?, ?)
    """, rule)

# ========== SAFETY WARNINGS ==========
warnings = [
    ("WARN-01", "temp > 205°C", "HIGH",
     "BENZENE FORMATION RISK: Temperature exceeds safe threshold",
     "Benzene, Methacrolein", "[8,42]"),

    ("WARN-02", "temp > 600°C", "CRITICAL",
     "COMBUSTION TOXICANTS: Full combustion generates carcinogens",
     "Benzene, Toluene, Naphthalene, PAHs", "[8,9]"),

    ("WARN-03", "Quercetin + Blood Thinners", "MEDIUM",
     "DRUG INTERACTION: Quercetin may potentiate anticoagulant effects",
     "Quercetin", "[37]"),

    ("WARN-04", "Terpinolene + Insomnia Target", "LOW",
     "PARADOXICAL EFFECT: Terpinolene may cause anxiety in sensitive users",
     "Terpinolene", "[15]"),

    ("WARN-05", "High THC + Anxiety History", "MEDIUM",
     "BIPHASIC RISK: THC may increase anxiety at high doses",
     "THC", "[23]"),

    ("WARN-06", "CBD + CYP450 Medications", "MEDIUM",
     "DRUG INTERACTION: CBD inhibits liver enzyme metabolism",
     "CBD", "[25]")
]

for warn in warnings:
    cursor.execute("""
        INSERT INTO safety_warnings
        (warning_id, trigger_condition, warning_level, message,
         compounds_involved, citation_refs)
        VALUES (?, ?, ?, ?, ?, ?)
    """, warn)
```

```
    # ========== TEMPERATURE PROFILES ==========
    temp_profiles = [
        ("Flavor Mode", 157, 170,
         "Volatile terpene extraction - Optimal flavor, minimal psychoactivity",
         "Pinene, THC (partial), Myrcene",
         "LOW", "Flavor enthusiasts, microdosing"),

        ("Balanced Mode", 170, 185,
         "Anxiolytic compounds activated - Best for anxiety management",
         "Limonene, CBD, Apigenin, THC",
         "LOW", "Anxiety, stress, mood disorders"),

        ("Medical Mode", 185, 200,
         "Analgesic and sedative compounds - Optimal for pain/sleep",
         "Linalool, Cannflavin A, CBN, CBD",
         "MEDIUM", "Pain, insomnia, inflammation"),

        ("Full Extraction", 200, 220,
         "High-boiling cannabinoids - Maximum therapeutic range",
         "THCV, CBC, β-Caryophyllene",
         "MEDIUM", "Appetite control, mood support"),

        ("High Risk Zone", 220, 600,
         "Quercetin activation but increased toxicant risk",
         "Quercetin, degradation products",
         "HIGH", "Not recommended - use lower temps"),

        ("Combustion Zone", 600, None,
         "TOXICITY ALERT - Benzene and carcinogen formation",
         "All compounds + toxicants",
         "CRITICAL", "AVOID - Use vaporization instead")
    ]

    for profile in temp_profiles:
        cursor.execute("""
            INSERT INTO temp_profiles
            (temp_range_name, temp_min, temp_max, description,
             active_compounds, risk_level, recommended_for)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        """, profile)

    self.conn.commit()
    print("✓ Pharmacognosy database populated with comprehensive chemical profiles")

# ========== QUERY METHODS ==========

def get_terpene(self, name: str) -> Optional[Dict]:
    """Retrieve terpene profile by name"""
    cursor = self.conn.cursor()
```

```
    cursor.execute("""
        SELECT * FROM terpenes WHERE name = ? OR id = ?
    """, (name, name))
    row = cursor.fetchone()
    if row:
        cols = [desc[0] for desc in cursor.description]
        return dict(zip(cols, row))
    return None

def get_cannabinoid(self, name: str) -> Optional[Dict]:
    """Retrieve cannabinoid profile by name"""
    cursor = self.conn.cursor()
    cursor.execute("""
        SELECT * FROM cannabinoids WHERE name = ? OR id = ?
    """, (name, name))
    row = cursor.fetchone()
    if row:
        cols = [desc[0] for desc in cursor.description]
```