```
-- Ephemeral Vault System Database Schema

-- ==================== SESSIONS TABLE ====================
CREATE TABLE ephemeral_sessions (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) UNIQUE NOT NULL,
    parent_wallet VARCHAR(44) NOT NULL,
    ephemeral_wallet VARCHAR(44) NOT NULL,
    encrypted_keypair BYTEA NOT NULL,
    vault_pda VARCHAR(44) NOT NULL,

    -- Timing
    created_at BIGINT NOT NULL,
    expires_at BIGINT NOT NULL,
    last_activity BIGINT NOT NULL DEFAULT EXTRACT(EPOCH FROM NOW()),
    revoked_at BIGINT,
    cleaned_at BIGINT,

    -- Status
    is_active BOOLEAN NOT NULL DEFAULT true,

    -- Financials
    approved_amount BIGINT NOT NULL,
    total_deposited BIGINT NOT NULL DEFAULT 0,
    total_spent BIGINT NOT NULL DEFAULT 0,

    -- Metadata
    user_ip VARCHAR(45),
    user_agent TEXT,
    device_fingerprint VARCHAR(64),

    created_at_timestamp TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_sessions_parent_wallet ON ephemeral_sessions(parent_wallet);
CREATE INDEX idx_sessions_session_id ON ephemeral_sessions(session_id);
CREATE INDEX idx_sessions_active ON ephemeral_sessions(is_active) WHERE is_active = true;
CREATE INDEX idx_sessions_expires_at ON ephemeral_sessions(expires_at);
CREATE INDEX idx_sessions_vault_pda ON ephemeral_sessions(vault_pda);

-- ==================== VAULT TRANSACTIONS TABLE ====================
CREATE TABLE vault_transactions (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) NOT NULL REFERENCES ephemeral_sessions(session_id),
    transaction_signature VARCHAR(88) NOT NULL UNIQUE,
    transaction_type VARCHAR(20) NOT NULL, -- 'deposit', 'trade', 'withdraw', 'revoke', 'cleanup'

    -- Transaction details
    amount BIGINT NOT NULL,
    fee_amount BIGINT,
    trade_id BIGINT,

    -- Status
    status VARCHAR(20) NOT NULL DEFAULT 'pending', -- 'pending', 'confirmed', 'failed'
    confirmed_at BIGINT,
```

```
    -- Blockchain info
    block_slot BIGINT,

    -- Metadata
    metadata JSONB,
    error_message TEXT,

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_transactions_session_id ON vault_transactions(session_id);
CREATE INDEX idx_transactions_signature ON vault_transactions(transaction_signature);
CREATE INDEX idx_transactions_type ON vault_transactions(transaction_type);
CREATE INDEX idx_transactions_status ON vault_transactions(status);
CREATE INDEX idx_transactions_created_at ON vault_transactions(created_at);

-- ==================== DELEGATION HISTORY TABLE ====================
CREATE TABLE delegation_history (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) NOT NULL REFERENCES ephemeral_sessions(session_id),
    vault_pda VARCHAR(44) NOT NULL,
    delegate_wallet VARCHAR(44) NOT NULL,

    -- Timing
    approved_at BIGINT NOT NULL,
    revoked_at BIGINT,

    -- Transaction info
    approval_signature VARCHAR(88),
    revocation_signature VARCHAR(88),

    -- Status
    is_active BOOLEAN NOT NULL DEFAULT true,

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_delegation_session_id ON delegation_history(session_id);
CREATE INDEX idx_delegation_vault_pda ON delegation_history(vault_pda);
CREATE INDEX idx_delegation_active ON delegation_history(is_active) WHERE is_active = true;

-- ==================== CLEANUP EVENTS TABLE ====================
CREATE TABLE cleanup_events (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) NOT NULL REFERENCES ephemeral_sessions(session_id),
    vault_pda VARCHAR(44) NOT NULL,

    -- Financial details
    returned_to_parent BIGINT NOT NULL,
    cleanup_reward BIGINT NOT NULL,

    -- Cleanup info
    cleanup_caller VARCHAR(44) NOT NULL,
    cleanup_signature VARCHAR(88) NOT NULL,
    cleanup_timestamp BIGINT NOT NULL,

    -- Reason
```

```
    cleanup_reason VARCHAR(50) NOT NULL, -- 'expired', 'manual_revoke', 'emergency'

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_cleanup_session_id ON cleanup_events(session_id);
CREATE INDEX idx_cleanup_vault_pda ON cleanup_events(vault_pda);
CREATE INDEX idx_cleanup_timestamp ON cleanup_events(cleanup_timestamp);

-- ==================== SESSION ANALYTICS TABLE ====================
CREATE TABLE session_analytics (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) NOT NULL REFERENCES ephemeral_sessions(session_id),

    -- Usage metrics
    total_trades INTEGER NOT NULL DEFAULT 0,
    successful_trades INTEGER NOT NULL DEFAULT 0,
    failed_trades INTEGER NOT NULL DEFAULT 0,

    -- Financial metrics
    total_volume BIGINT NOT NULL DEFAULT 0,
    total_fees_paid BIGINT NOT NULL DEFAULT 0,
    avg_trade_size BIGINT,

    -- Performance metrics
    avg_confirmation_time INTEGER, -- milliseconds
    session_duration INTEGER, -- seconds

    -- Activity pattern
    trades_per_minute NUMERIC(10, 2),
    peak_activity_hour INTEGER,

    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_analytics_session_id ON session_analytics(session_id);

-- ==================== SECURITY ALERTS TABLE ====================
CREATE TABLE security_alerts (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36) REFERENCES ephemeral_sessions(session_id),
    alert_type VARCHAR(30) NOT NULL, -- 'unusual_spending', 'high_frequency', 'ip_change', 'expired_session'
    severity VARCHAR(10) NOT NULL, -- 'low', 'medium', 'high', 'critical'

    -- Alert details
    description TEXT NOT NULL,
    metadata JSONB,

    -- Status
    is_resolved BOOLEAN NOT NULL DEFAULT false,
    resolved_at TIMESTAMP,
    resolved_by VARCHAR(100),

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_alerts_session_id ON security_alerts(session_id);
```

```
CREATE INDEX idx_alerts_type ON security_alerts(alert_type);
CREATE INDEX idx_alerts_severity ON security_alerts(severity);
CREATE INDEX idx_alerts_unresolved ON security_alerts(is_resolved) WHERE is_resolved = false;

-- ==================== RATE LIMITING TABLE ====================
CREATE TABLE rate_limits (
    id SERIAL PRIMARY KEY,
    parent_wallet VARCHAR(44) NOT NULL,
    action_type VARCHAR(30) NOT NULL, -- 'session_create', 'deposit', 'trade'

    -- Rate limit tracking
    request_count INTEGER NOT NULL DEFAULT 1,
    window_start BIGINT NOT NULL,
    window_end BIGINT NOT NULL,

    -- Status
    is_blocked BOOLEAN NOT NULL DEFAULT false,
    blocked_until BIGINT,

    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_rate_limits_wallet ON rate_limits(parent_wallet);
CREATE INDEX idx_rate_limits_action ON rate_limits(action_type);
CREATE INDEX idx_rate_limits_window ON rate_limits(window_end);

-- ==================== AUDIT LOG TABLE ====================
CREATE TABLE audit_log (
    id SERIAL PRIMARY KEY,
    session_id VARCHAR(36),
    parent_wallet VARCHAR(44),

    -- Action details
    action VARCHAR(50) NOT NULL,
    actor VARCHAR(44) NOT NULL, -- wallet that performed action
    target VARCHAR(44), -- target wallet/account if applicable

    -- Request details
    ip_address VARCHAR(45),
    user_agent TEXT,

    -- Result
    success BOOLEAN NOT NULL,
    error_message TEXT,

    -- Additional data
    metadata JSONB,

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_audit_session_id ON audit_log(session_id);
CREATE INDEX idx_audit_parent_wallet ON audit_log(parent_wallet);
CREATE INDEX idx_audit_action ON audit_log(action);
CREATE INDEX idx_audit_created_at ON audit_log(created_at);
```

```
-- ==================== VIEWS ====================

-- Active sessions overview
CREATE VIEW active_sessions_view AS
SELECT
    s.session_id,
    s.parent_wallet,
    s.vault_pda,
    s.created_at,
    s.expires_at,
    s.total_deposited,
    s.total_spent,
    (s.total_deposited - s.total_spent) as remaining_balance,
    a.total_trades,
    a.total_volume
FROM ephemeral_sessions s
LEFT JOIN session_analytics a ON s.session_id = a.session_id
WHERE s.is_active = true;

-- Sessions needing cleanup
CREATE VIEW cleanup_candidates_view AS
SELECT
    session_id,
    parent_wallet,
    vault_pda,
    expires_at,
    total_deposited,
    total_spent,
    EXTRACT(EPOCH FROM NOW()) - expires_at as seconds_expired
FROM ephemeral_sessions
WHERE (expires_at < EXTRACT(EPOCH FROM NOW()) OR is_active = false)
  AND cleaned_at IS NULL;

-- Security alerts summary
CREATE VIEW alerts_summary_view AS
SELECT
    s.session_id,
    s.parent_wallet,
    COUNT(a.id) as total_alerts,
    SUM(CASE WHEN a.severity = 'critical' THEN 1 ELSE 0 END) as critical_alerts,
    SUM(CASE WHEN a.severity = 'high' THEN 1 ELSE 0 END) as high_alerts,
    MAX(a.created_at) as last_alert_time
FROM ephemeral_sessions s
LEFT JOIN security_alerts a ON s.session_id = a.session_id
WHERE a.is_resolved = false
GROUP BY s.session_id, s.parent_wallet;

-- ==================== FUNCTIONS ====================

-- Function to update session activity
CREATE OR REPLACE FUNCTION update_session_activity()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE ephemeral_sessions
    SET last_activity = EXTRACT(EPOCH FROM NOW()),
        updated_at = NOW()
    WHERE session_id = NEW.session_id;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger for transaction activity
CREATE TRIGGER update_activity_on_transaction
AFTER INSERT ON vault_transactions
FOR EACH ROW
EXECUTE FUNCTION update_session_activity();

-- Function to update analytics
CREATE OR REPLACE FUNCTION update_session_analytics()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO session_analytics (session_id, total_trades, successful_trades, failed_trades)
    VALUES (NEW.session_id, 0, 0, 0)
    ON CONFLICT (session_id) DO NOTHING;

    IF NEW.transaction_type = 'trade' THEN
        UPDATE session_analytics
        SET total_trades = total_trades + 1,
            successful_trades = CASE WHEN NEW.status = 'confirmed' THEN successful_trades + 1 ELSE successful_trades END,
            failed_trades = CASE WHEN NEW.status = 'failed' THEN failed_trades + 1 ELSE failed_trades END,
            total_volume = total_volume + NEW.amount,
            total_fees_paid = total_fees_paid + COALESCE(NEW.fee_amount, 0),
            updated_at = NOW()
        WHERE session_id = NEW.session_id;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger for analytics update
CREATE TRIGGER update_analytics_on_transaction
AFTER INSERT OR UPDATE ON vault_transactions
FOR EACH ROW
EXECUTE FUNCTION update_session_analytics();

-- ==================== INITIAL DATA ====================

-- Insert sample rate limit configurations
INSERT INTO rate_limits (parent_wallet, action_type, request_count, window_start, window_end)
VALUES
    ('system_default', 'session_create', 0, EXTRACT(EPOCH FROM NOW()), EXTRACT(EPOCH FROM NOW()) + 3600),
    ('system_default', 'deposit', 0, EXTRACT(EPOCH FROM NOW()), EXTRACT(EPOCH FROM NOW()) + 3600),
    ('system_default', 'trade', 0, EXTRACT(EPOCH FROM NOW()), EXTRACT(EPOCH FROM NOW()) + 60);

-- Grant permissions (adjust as needed)
-- GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO vault_service_user;
-- GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO vault_service_user;
```