-- ============================================================================

-- INTERPAY PAYMENT SYSTEM - DATABASE SCHEMA

-- Database-per-Service Pattern with PostgreSQL, Cassandra, and Redis

-- ============================================================================

-- ============================================================================

-- 1. USER SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE users (

user\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20) UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE', -- ACTIVE, SUSPENDED, CLOSED

account\_type VARCHAR(20) NOT NULL, -- PERSONAL, BUSINESS, MERCHANT

kyc\_status VARCHAR(20) DEFAULT 'PENDING', -- PENDING, VERIFIED, REJECTED

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

last\_login\_at TIMESTAMP,

INDEX idx\_email (email),

INDEX idx\_phone (phone),

INDEX idx\_status (status)

);

CREATE TABLE user\_profiles (

profile\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL UNIQUE,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

date\_of\_birth DATE,

country\_code VARCHAR(3) NOT NULL,

address\_line1 VARCHAR(255),

address\_line2 VARCHAR(255),

city VARCHAR(100),

state VARCHAR(100),

postal\_code VARCHAR(20),

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

INDEX idx\_user\_id (user\_id)

);

CREATE TABLE kyc\_documents (

document\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL,

document\_type VARCHAR(50) NOT NULL, -- PASSPORT, DRIVERS\_LICENSE, NATIONAL\_ID

document\_number VARCHAR(100) NOT NULL,

document\_url VARCHAR(500),

verification\_status VARCHAR(20) DEFAULT 'PENDING',

verified\_at TIMESTAMP,

verified\_by UUID,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

INDEX idx\_user\_id (user\_id),

INDEX idx\_status (verification\_status)

);

-- ============================================================================

-- 2. WALLET SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE wallets (

wallet\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL UNIQUE,

currency\_code VARCHAR(3) NOT NULL DEFAULT 'USD',

balance DECIMAL(20, 4) NOT NULL DEFAULT 0.0000,

available\_balance DECIMAL(20, 4) NOT NULL DEFAULT 0.0000,

pending\_balance DECIMAL(20, 4) NOT NULL DEFAULT 0.0000,

status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE', -- ACTIVE, FROZEN, CLOSED

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

version INTEGER NOT NULL DEFAULT 1, -- For optimistic locking

INDEX idx\_user\_id (user\_id),

INDEX idx\_status (status),

CONSTRAINT balance\_non\_negative CHECK (balance >= 0),

CONSTRAINT available\_balance\_non\_negative CHECK (available\_balance >= 0)

);

CREATE TABLE wallet\_transactions (

transaction\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

wallet\_id UUID NOT NULL,

transaction\_type VARCHAR(30) NOT NULL, -- CREDIT, DEBIT, HOLD, RELEASE

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

balance\_before DECIMAL(20, 4) NOT NULL,

balance\_after DECIMAL(20, 4) NOT NULL,

reference\_id UUID, -- Links to payment or transfer

reference\_type VARCHAR(50), -- PAYMENT, TRANSFER, REFUND, FEE

description TEXT,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (wallet\_id) REFERENCES wallets(wallet\_id),

INDEX idx\_wallet\_id (wallet\_id),

INDEX idx\_reference (reference\_id, reference\_type),

INDEX idx\_created\_at (created\_at),

INDEX idx\_transaction\_type (transaction\_type)

);

CREATE TABLE wallet\_holds (

hold\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

wallet\_id UUID NOT NULL,

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

reference\_id UUID NOT NULL,

reference\_type VARCHAR(50) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE', -- ACTIVE, RELEASED, CAPTURED

expires\_at TIMESTAMP NOT NULL,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

released\_at TIMESTAMP,

FOREIGN KEY (wallet\_id) REFERENCES wallets(wallet\_id),

INDEX idx\_wallet\_id (wallet\_id),

INDEX idx\_status (status),

INDEX idx\_reference (reference\_id),

INDEX idx\_expires\_at (expires\_at)

);

-- ============================================================================

-- 3. PAYMENT SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE payments (

payment\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

merchant\_id UUID NOT NULL,

customer\_id UUID, -- NULL for guest payments

payment\_method VARCHAR(30) NOT NULL, -- WALLET, CARD, GUEST

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL DEFAULT 'USD',

status VARCHAR(30) NOT NULL DEFAULT 'INITIATED', -- INITIATED, PENDING, AUTHORIZED, CAPTURED, FAILED, CANCELLED, REFUNDED

payment\_gateway VARCHAR(50), -- INTERPAY, STRIPE\_FALLBACK

merchant\_order\_id VARCHAR(100),

description TEXT,

metadata JSONB,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

completed\_at TIMESTAMP,

INDEX idx\_merchant\_id (merchant\_id),

INDEX idx\_customer\_id (customer\_id),

INDEX idx\_status (status),

INDEX idx\_created\_at (created\_at),

INDEX idx\_merchant\_order (merchant\_id, merchant\_order\_id)

);

CREATE TABLE payment\_methods (

payment\_method\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL,

method\_type VARCHAR(30) NOT NULL, -- CARD, BANK\_ACCOUNT, WALLET

is\_default BOOLEAN DEFAULT FALSE,

status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_user\_id (user\_id),

INDEX idx\_status (status)

);

CREATE TABLE cards (

card\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

payment\_method\_id UUID NOT NULL UNIQUE,

user\_id UUID NOT NULL,

card\_token VARCHAR(255) NOT NULL, -- Tokenized card number

card\_brand VARCHAR(20) NOT NULL, -- VISA, MASTERCARD, AMEX

last\_four VARCHAR(4) NOT NULL,

expiry\_month INTEGER NOT NULL,

expiry\_year INTEGER NOT NULL,

cardholder\_name VARCHAR(200),

billing\_address\_id UUID,

is\_verified BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (payment\_method\_id) REFERENCES payment\_methods(payment\_method\_id) ON DELETE CASCADE,

INDEX idx\_user\_id (user\_id),

INDEX idx\_card\_token (card\_token)

);

CREATE TABLE payment\_transactions (

transaction\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

payment\_id UUID NOT NULL,

transaction\_type VARCHAR(30) NOT NULL, -- AUTHORIZE, CAPTURE, VOID, REFUND

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

status VARCHAR(20) NOT NULL, -- SUCCESS, FAILED, PENDING

gateway\_transaction\_id VARCHAR(255),

gateway\_response JSONB,

error\_code VARCHAR(50),

error\_message TEXT,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (payment\_id) REFERENCES payments(payment\_id),

INDEX idx\_payment\_id (payment\_id),

INDEX idx\_status (status),

INDEX idx\_created\_at (created\_at)

);

CREATE TABLE refunds (

refund\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

payment\_id UUID NOT NULL,

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

reason VARCHAR(100),

status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- PENDING, COMPLETED, FAILED

initiated\_by UUID,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

completed\_at TIMESTAMP,

FOREIGN KEY (payment\_id) REFERENCES payments(payment\_id),

INDEX idx\_payment\_id (payment\_id),

INDEX idx\_status (status)

);

-- ============================================================================

-- 4. TRANSFER SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE transfers (

transfer\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

sender\_wallet\_id UUID NOT NULL,

receiver\_wallet\_id UUID NOT NULL,

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

fee\_amount DECIMAL(20, 4) DEFAULT 0.0000,

status VARCHAR(30) NOT NULL DEFAULT 'INITIATED', -- INITIATED, PENDING, COMPLETED, FAILED, REVERSED

transfer\_type VARCHAR(30) NOT NULL, -- P2P, PAYOUT, REFUND

description TEXT,

metadata JSONB,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

completed\_at TIMESTAMP,

INDEX idx\_sender\_wallet (sender\_wallet\_id),

INDEX idx\_receiver\_wallet (receiver\_wallet\_id),

INDEX idx\_status (status),

INDEX idx\_created\_at (created\_at)

);

-- ============================================================================

-- 5. MERCHANT SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE merchants (

merchant\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL UNIQUE,

business\_name VARCHAR(255) NOT NULL,

business\_type VARCHAR(50) NOT NULL,

business\_category VARCHAR(100),

website\_url VARCHAR(500),

tax\_id VARCHAR(50),

status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- PENDING, APPROVED, SUSPENDED, REJECTED

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

approved\_at TIMESTAMP,

INDEX idx\_user\_id (user\_id),

INDEX idx\_status (status)

);

CREATE TABLE merchant\_api\_keys (

api\_key\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

merchant\_id UUID NOT NULL,

key\_name VARCHAR(100) NOT NULL,

public\_key VARCHAR(255) UNIQUE NOT NULL,

secret\_key\_hash VARCHAR(255) NOT NULL,

environment VARCHAR(20) NOT NULL, -- SANDBOX, PRODUCTION

status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',

permissions JSONB,

last\_used\_at TIMESTAMP,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP,

FOREIGN KEY (merchant\_id) REFERENCES merchants(merchant\_id) ON DELETE CASCADE,

INDEX idx\_merchant\_id (merchant\_id),

INDEX idx\_public\_key (public\_key)

);

CREATE TABLE merchant\_settlements (

settlement\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

merchant\_id UUID NOT NULL,

amount DECIMAL(20, 4) NOT NULL,

currency\_code VARCHAR(3) NOT NULL,

fee\_amount DECIMAL(20, 4) NOT NULL,

net\_amount DECIMAL(20, 4) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- PENDING, COMPLETED, FAILED

settlement\_date DATE NOT NULL,

bank\_reference VARCHAR(100),

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

completed\_at TIMESTAMP,

FOREIGN KEY (merchant\_id) REFERENCES merchants(merchant\_id),

INDEX idx\_merchant\_id (merchant\_id),

INDEX idx\_settlement\_date (settlement\_date),

INDEX idx\_status (status)

);

-- ============================================================================

-- 6. FRAUD DETECTION SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE fraud\_rules (

rule\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

rule\_name VARCHAR(100) NOT NULL,

rule\_type VARCHAR(50) NOT NULL, -- VELOCITY, AMOUNT, LOCATION, DEVICE, PATTERN

conditions JSONB NOT NULL,

action VARCHAR(30) NOT NULL, -- BLOCK, REVIEW, ALLOW\_WITH\_MFA

priority INTEGER NOT NULL DEFAULT 100,

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_rule\_type (rule\_type),

INDEX idx\_priority (priority)

);

CREATE TABLE fraud\_checks (

check\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

entity\_id UUID NOT NULL, -- payment\_id, transfer\_id, etc

entity\_type VARCHAR(50) NOT NULL,

risk\_score DECIMAL(5, 2) NOT NULL,

risk\_level VARCHAR(20) NOT NULL, -- LOW, MEDIUM, HIGH, CRITICAL

decision VARCHAR(30) NOT NULL, -- APPROVED, DECLINED, REVIEW

triggered\_rules JSONB,

ip\_address INET,

device\_fingerprint VARCHAR(255),

user\_agent TEXT,

geolocation JSONB,

checked\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_entity (entity\_id, entity\_type),

INDEX idx\_risk\_level (risk\_level),

INDEX idx\_decision (decision),

INDEX idx\_checked\_at (checked\_at)

);

CREATE TABLE blacklist (

blacklist\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

entity\_type VARCHAR(50) NOT NULL, -- EMAIL, IP, CARD, DEVICE

entity\_value VARCHAR(500) NOT NULL,

reason TEXT,

added\_by UUID,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP,

INDEX idx\_entity (entity\_type, entity\_value)

);

-- ============================================================================

-- 7. NOTIFICATION SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE notifications (

notification\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID NOT NULL,

notification\_type VARCHAR(50) NOT NULL, -- EMAIL, SMS, PUSH, IN\_APP

template\_id VARCHAR(100) NOT NULL,

subject VARCHAR(255),

content TEXT NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'PENDING', -- PENDING, SENT, FAILED

priority VARCHAR(20) DEFAULT 'NORMAL',

metadata JSONB,

scheduled\_at TIMESTAMP,

sent\_at TIMESTAMP,

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_user\_id (user\_id),

INDEX idx\_status (status),

INDEX idx\_scheduled\_at (scheduled\_at)

);

-- ============================================================================

-- 8. AUDIT SERVICE DATABASE (PostgreSQL)

-- ============================================================================

CREATE TABLE audit\_logs (

audit\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID,

service\_name VARCHAR(50) NOT NULL,

action VARCHAR(100) NOT NULL,

entity\_type VARCHAR(50),

entity\_id UUID,

old\_values JSONB,

new\_values JSONB,

ip\_address INET,

user\_agent TEXT,

status VARCHAR(20), -- SUCCESS, FAILURE

created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_user\_id (user\_id),

INDEX idx\_entity (entity\_type, entity\_id),

INDEX idx\_created\_at (created\_at),

INDEX idx\_service (service\_name)

);

-- ============================================================================

-- CASSANDRA SCHEMAS (CQL)

-- ============================================================================

-- Session Store

CREATE KEYSPACE IF NOT EXISTS interpay\_sessions

WITH replication = {'class': 'NetworkTopologyStrategy', 'datacenter1': 3};

CREATE TABLE interpay\_sessions.user\_sessions (

session\_id UUID,

user\_id UUID,

device\_id TEXT,

ip\_address TEXT,

user\_agent TEXT,

created\_at TIMESTAMP,

last\_activity TIMESTAMP,

expires\_at TIMESTAMP,

session\_data MAP<TEXT, TEXT>,

PRIMARY KEY (session\_id)

) WITH default\_time\_to\_live = 86400

AND gc\_grace\_seconds = 86400;

CREATE INDEX ON interpay\_sessions.user\_sessions (user\_id);

-- Activity Logs

CREATE KEYSPACE IF NOT EXISTS interpay\_logs

WITH replication = {'class': 'NetworkTopologyStrategy', 'datacenter1': 3};

CREATE TABLE interpay\_logs.payment\_logs (

payment\_id UUID,

log\_timestamp TIMESTAMP,

log\_level TEXT,

service\_name TEXT,

message TEXT,

metadata MAP<TEXT, TEXT>,

PRIMARY KEY (payment\_id, log\_timestamp)

) WITH CLUSTERING ORDER BY (log\_timestamp DESC)

AND default\_time\_to\_live = 2592000; -- 30 days

CREATE TABLE interpay\_logs.api\_request\_logs (

request\_id UUID,

timestamp TIMESTAMP,

merchant\_id UUID,

endpoint TEXT,

method TEXT,

request\_body TEXT,

response\_body TEXT,

response\_code INT,

duration\_ms INT,

ip\_address TEXT,

PRIMARY KEY ((merchant\_id), timestamp, request\_id)

) WITH CLUSTERING ORDER BY (timestamp DESC)

AND default\_time\_to\_live = 7776000; -- 90 days

CREATE TABLE interpay\_logs.fraud\_events (

event\_id UUID,

timestamp TIMESTAMP,

user\_id UUID,

event\_type TEXT,

risk\_score DOUBLE,

details MAP<TEXT, TEXT>,

PRIMARY KEY ((user\_id), timestamp, event\_id)

) WITH CLUSTERING ORDER BY (timestamp DESC);

-- Transaction Analytics (Time-series data)

CREATE TABLE interpay\_logs.transaction\_metrics (

metric\_date DATE,

hour INT,

merchant\_id UUID,

transaction\_count COUNTER,

total\_amount COUNTER,

successful\_count COUNTER,

failed\_count COUNTER,

PRIMARY KEY ((metric\_date, hour), merchant\_id)

);

-- ============================================================================

-- REDIS CACHE STRUCTURES (Pseudo-schema for documentation)

-- ============================================================================

-- Key Patterns:

-- user:{user\_id}:profile - User profile cache (TTL: 1 hour)

-- wallet:{wallet\_id}:balance - Wallet balance cache (TTL: 5 minutes)

-- payment:{payment\_id}:status - Payment status cache (TTL: 10 minutes)

-- session:{session\_id} - User session data (TTL: 24 hours)

-- fraud:velocity:{user\_id}:{timeframe} - Velocity check counters (TTL: variable)

-- api:ratelimit:{merchant\_id}:{minute} - API rate limiting (TTL: 1 minute)

-- merchant:{merchant\_id}:apikey:{key} - API key validation cache (TTL: 1 hour)

-- card:{card\_token}:metadata - Card metadata cache (TTL: 30 minutes)

-- transfer:{transfer\_id}:lock - Distributed lock for transfers (TTL: 30 seconds)

-- Redis Data Structures Used:

-- STRING: Simple key-value for caching

-- HASH: Complex objects like user profiles

-- SORTED SET: Leaderboards, recent transactions

-- SET: Unique collections (blacklisted IPs)

-- LIST: Recent activity feeds

-- BITMAP: Daily active users tracking

-- HyperLogLog: Unique visitor counts

-- ============================================================================

-- INDEXES AND PERFORMANCE OPTIMIZATION

-- ============================================================================

-- Additional composite indexes for common query patterns

CREATE INDEX idx\_payments\_merchant\_status\_created ON payments(merchant\_id, status, created\_at);

CREATE INDEX idx\_wallet\_transactions\_wallet\_created ON wallet\_transactions(wallet\_id, created\_at DESC);

CREATE INDEX idx\_transfers\_status\_created ON transfers(status, created\_at);

CREATE INDEX idx\_fraud\_checks\_risk\_checked ON fraud\_checks(risk\_level, checked\_at);

-- Partitioning strategy for large tables (PostgreSQL 12+)

-- Partition wallet\_transactions by created\_at (monthly)

-- Partition audit\_logs by created\_at (monthly)

-- Partition payment\_transactions by created\_at (monthly)

-- ============================================================================

-- MATERIALIZED VIEWS FOR ANALYTICS

-- ============================================================================

CREATE MATERIALIZED VIEW merchant\_daily\_summary AS

SELECT

merchant\_id,

DATE(created\_at) as transaction\_date,

COUNT(\*) as transaction\_count,

SUM(amount) as total\_amount,

SUM(CASE WHEN status = 'CAPTURED' THEN amount ELSE 0 END) as successful\_amount,

COUNT(CASE WHEN status = 'FAILED' THEN 1 END) as failed\_count

FROM payments

GROUP BY merchant\_id, DATE(created\_at)

WITH DATA;

CREATE UNIQUE INDEX ON merchant\_daily\_summary(merchant\_id, transaction\_date);

-- ============================================================================

-- TRIGGERS FOR AUDIT AND CONSISTENCY

-- ============================================================================

CREATE OR REPLACE FUNCTION update\_timestamp()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_users\_timestamp

BEFORE UPDATE ON users

FOR EACH ROW EXECUTE FUNCTION update\_timestamp();

CREATE TRIGGER update\_wallets\_timestamp

BEFORE UPDATE ON wallets

FOR EACH ROW EXECUTE FUNCTION update\_timestamp();

CREATE TRIGGER update\_payments\_timestamp

BEFORE UPDATE ON payments

FOR EACH ROW EXECUTE FUNCTION update\_timestamp();