🗄️ Referência de Schema do Banco de Dados

Tabelas Existentes (Atual)

1. profiles

```sql
CREATE TABLE profiles (
  id UUID PRIMARY KEY REFERENCES auth.users(id),
  email TEXT,
  name TEXT,
  avatar_url TEXT,
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);
```

2. user_settings

```sql
CREATE TABLE user_settings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  bot_status TEXT CHECK (bot_status IN ('stopped', 'running', 'paused')),
  paper_mode BOOLEAN DEFAULT true,
  balance NUMERIC DEFAULT 10000,
  risk_per_trade NUMERIC DEFAULT 2.0,
  leverage INTEGER,
  max_positions INTEGER,
  profit_target_percent NUMERIC,
  active_strategies TEXT[],
  trading_strategy TEXT,
  single_position_mode BOOLEAN,
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);
```

3. active_positions

```sql
CREATE TABLE active_positions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  asset TEXT NOT NULL,
  direction TEXT NOT NULL CHECK (direction IN ('LONG', 'SHORT')),
  entry_price NUMERIC NOT NULL,
  stop_loss NUMERIC NOT NULL,
  take_profit NUMERIC NOT NULL,
  risk_reward NUMERIC NOT NULL,
  current_price NUMERIC,
  current_pnl NUMERIC,
  projected_profit NUMERIC NOT NULL,
  session TEXT,
  agents JSONB,  -- 👈 Campo para armazenar info do Vision Agent
  opened_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);
```

4. operations

```
CREATE TABLE operations (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  asset TEXT NOT NULL,
  direction TEXT NOT NULL,
  entry_price NUMERIC NOT NULL,
  exit_price NUMERIC NOT NULL,
  stop_loss NUMERIC NOT NULL,
  take_profit NUMERIC NOT NULL,
  risk_reward NUMERIC NOT NULL,
  pnl NUMERIC NOT NULL,
  profit_percent NUMERIC NOT NULL,
  result TEXT CHECK (result IN ('WIN', 'LOSS')),
  entry_time TIMESTAMPTZ NOT NULL,
  exit_time TIMESTAMPTZ NOT NULL,
  session TEXT,
  agents JSONB,  -- 👈 Campo para armazenar info do Vision Agent
  created_at TIMESTAMPTZ DEFAULT now()
);
```

5. pending_signals

```
CREATE TABLE pending_signals (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  asset TEXT NOT NULL,
  signal_type TEXT CHECK (signal_type IN ('ENTER', 'EXIT')),
  direction TEXT CHECK (direction IN ('LONG', 'SHORT')),
  entry_price NUMERIC,
  stop_loss NUMERIC,
  take_profit NUMERIC,
  risk_reward NUMERIC,
  confidence NUMERIC,  -- 👈 Confidence do modelo ML
  signal_data JSONB,   -- 👈 Dados extras (video_id, model_version, etc)
  status TEXT CHECK (status IN ('pending', 'executed', 'cancelled')),
  created_at TIMESTAMPTZ DEFAULT now(),
  executed_at TIMESTAMPTZ
);
```

6. agent_logs

```
CREATE TABLE agent_logs (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  agent_name TEXT NOT NULL,  -- 👈 'vision_trading_agent'
  action TEXT NOT NULL,      -- 👈 'signal_enter', 'signal_exit', 'video_processed'
  status TEXT NOT NULL,      -- 👈 'success', 'failed', 'pending'
  details JSONB,             -- 👈 Informações detalhadas
  created_at TIMESTAMPTZ DEFAULT now()
);
```

7. user_api_credentials

```
CREATE TABLE user_api_credentials (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id),
  broker_type TEXT CHECK (broker_type IN ('binance', 'forex')),
  encrypted_api_key TEXT,
  encrypted_api_secret TEXT,
  broker_name TEXT,
  is_active BOOLEAN DEFAULT true,
  last_tested_at TIMESTAMPTZ,
  test_status TEXT CHECK (test_status IN ('success', 'failed', 'pending')),
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now(),
  UNIQUE(user_id, broker_type)
);
```

Tabelas a Criar (Vision Agent)

8. vision_agent_videos ✨ NOVA

```
CREATE TABLE vision_agent_videos (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  video_id TEXT NOT NULL,        -- ID do YouTube
  youtube_url TEXT NOT NULL,
  title TEXT,
  channel TEXT,
  status TEXT CHECK (status IN ('pending', 'processing', 'completed', 'failed')) DE-
FAULT 'pending',
  total_frames INTEGER,
  processed_frames INTEGER DEFAULT 0,
  signals_generated INTEGER DEFAULT 0,
  model_version TEXT,
  processing_started_at TIMESTAMPTZ,
  processing_completed_at TIMESTAMPTZ,
  error_message TEXT,
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);

CREATE INDEX idx_vision_agent_videos_user_id ON vision_agent_videos(user_id);
CREATE INDEX idx_vision_agent_videos_status ON vision_agent_videos(status);
```

9. vision_agent_settings ✨ NOVA

```sql
CREATE TABLE vision_agent_settings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL UNIQUE,
  enabled BOOLEAN DEFAULT false,
  mode TEXT CHECK (mode IN ('SHADOW', 'PAPER', 'LIVE')) DEFAULT 'SHADOW',
  confidence_threshold NUMERIC(3,2) DEFAULT 0.70,  -- 0.00 a 1.00
  youtube_playlist_url TEXT,
  model_version TEXT DEFAULT 'model_seq_v20251125.h5',
  auto_process_new_videos BOOLEAN DEFAULT false,
  max_signals_per_day INTEGER DEFAULT 50,
  cooldown_seconds INTEGER DEFAULT 300,  -- 5 minutos entre sinais do mesmo asset
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);
```

Estrutura JSON dos Campos JSONB

Campo `agents` em `active_positions` e `operations`

```json
{
  "source": "vision_trading_agent",
  "video_id": "dQw4w9WgXcQ",
  "video_title": "Como identificar Order Block em WIN",
  "confidence": 0.82,
  "model_version": "model_seq_v20251125.h5",
  "frame_index": 2400,
  "timestamp": "2025-11-25T12:45:32Z",
  "features": {
    "hands_detected": 1,
    "draw_count": 2,
    "ocr_text": "Entry 134.50",
    "arrows_detected": 1
  }
}
```

Campo `signal_data` em `pending_signals`

```json
{
  "source": "vision_trading_agent",
  "video_id": "dQw4w9WgXcQ",
  "model_version": "model_seq_v20251125.h5",
  "features_summary": {
    "hands": 1,
    "draw_count": 2,
    "ocr": "1.3450"
  },
  "raw_confidence": {
    "enter": 0.82,
    "exit": 0.12,
    "ignore": 0.06
  }
}
```

Campo `details` em `agent_logs`

```
{
  "signal_id": "uuid-do-sinal",
  "video_id": "dQw4w9WgXcQ",
  "confidence": 0.82,
  "action_taken": "signal_sent",
  "execution_time_ms": 150,
  "model_version": "model_seq_v20251125.h5"
}
```

Queries Úteis

Verificar últimos sinais do Vision Agent

```
SELECT
  ps.id,
  ps.asset,
  ps.signal_type,
  ps.confidence,
  ps.signal_data->>'video_id' as video_id,
  ps.status,
  ps.created_at
FROM pending_signals ps
WHERE ps.signal_data->>'source' = 'vision_trading_agent'
  AND ps.user_id = 'USER_UUID'
ORDER BY ps.created_at DESC
LIMIT 10;
```

Verificar vídeos processados

```
SELECT
  video_id,
  title,
  status,
  signals_generated,
  processing_completed_at
FROM vision_agent_videos
WHERE user_id = 'USER_UUID'
ORDER BY created_at DESC;
```

Verificar posições originadas pelo Vision Agent

```
SELECT
  ap.asset,
  ap.direction,
  ap.entry_price,
  ap.current_pnl,
  ap.agents->>'confidence' as confidence,
  ap.agents->>'video_id' as video_id
FROM active_positions ap
WHERE ap.agents->>'source' = 'vision_trading_agent'
  AND ap.user_id = 'USER_UUID';
```

Estatísticas de performance do Vision Agent

```sql
SELECT
  COUNT(*) as total_trades,
  SUM(CASE WHEN result = 'WIN' THEN 1 ELSE 0 END) as wins,
  SUM(CASE WHEN result = 'LOSS' THEN 1 ELSE 0 END) as losses,
  ROUND(AVG(pnl), 2) as avg_pnl,
  ROUND(SUM(pnl), 2) as total_pnl,
  ROUND(AVG(CAST(agents->>'confidence' AS NUMERIC)), 2) as avg_confidence
FROM operations
WHERE agents->>'source' = 'vision_trading_agent'
  AND user_id = 'USER_UUID'
  AND created_at >= CURRENT_DATE - INTERVAL '30 days';
```

Edge Functions

Existentes

1. **analyze-multi-timeframe**
   - Análise SMC em múltiplos timeframes
   - Detecta FVG, OB, Liquidity Sweeps

2. **execute-order**
   - Executa ordem de trading
   - Validações de risk management
   - Suporta paper_mode e modo real

3. **close-position**
   - Fecha posição manualmente
   - Registra em operations

4. **monitor-positions**
   - Monitora posições abertas
   - Atualiza PnL em tempo real
   - Fecha quando TP/SL atingido

5. **sync-real-balance**
   - Sincroniza saldo real da corretora

6. **test-broker-connection**
   - Testa conexão com Binance/Forex

7. **encrypt-api-credentials**
   - Criptografa credenciais da API

A Criar

1. **vision-agent-signal** ✨ NOVA
   - Recebe sinais do Vision Agent
   - Valida e armazena em pending_signals
   - Triggera execute-order se necessário
   - Registra logs

RLS (Row Level Security)

Todas as tabelas têm RLS habilitado com as seguintes policies:

```sql
-- Exemplo para vision_agent_videos
CREATE POLICY "Users can view their own videos"
ON vision_agent_videos FOR SELECT
TO authenticated
USING (auth.uid() = user_id);

CREATE POLICY "Users can insert their own videos"
ON vision_agent_videos FOR INSERT
TO authenticated
WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update their own videos"
ON vision_agent_videos FOR UPDATE
TO authenticated
USING (auth.uid() = user_id);

CREATE POLICY "Users can delete their own videos"
ON vision_agent_videos FOR DELETE
TO authenticated
USING (auth.uid() = user_id);
```

Documento: Referência de Schema
Versão: 1.0
Data: 25 de Novembro de 2025