Guia de Estrutura de Dados – Bot Cripto Telegram

# 1. Introdução

Este documento consolida a modelagem de dados do Bot Cripto Telegram. Inclui tabelas e funções no Supabase/Postgres, bem como a estratégia de armazenamento em Redis.

# 2. Estrutura no Supabase / PostgreSQL

### Tabelas Principais:

- users  
 - id (uuid, pk)  
 - chat\_id (bigint, único)  
 - created\_at (timestamp)  
 - updated\_at (timestamp)  
  
- alerts  
 - id (uuid, pk)  
 - user\_id (fk → users.id)  
 - symbol (text)  
 - direction (enum: above/below)  
 - target\_brl (numeric)  
 - created\_at (timestamp)  
 - triggered\_at (timestamp, nullable)  
  
- trades  
 - id (uuid, pk)  
 - user\_id (fk → users.id)  
 - symbol (text)  
 - side (enum: buy/sell)  
 - qty (numeric)  
 - price\_brl (numeric)  
 - fee\_brl (numeric)  
 - created\_at (timestamp)  
  
- bot\_log  
 - id (uuid, pk)  
 - user\_id (fk → users.id, nullable)  
 - action (text)  
 - payload (jsonb)  
 - created\_at (timestamp)  
  
- error\_state  
 - id (uuid, pk)  
 - workflow (text)  
 - error\_message (text)  
 - payload (jsonb)  
 - created\_at (timestamp)  
  
- api\_usage  
 - id (uuid, pk)  
 - api\_name (text)  
 - call\_count (int)  
 - last\_called\_at (timestamp)

### Funções Auxiliares:

- fn\_positions\_cma(user\_id): calcula posições e custo médio.   
- fn\_pnl\_cma(user\_id): calcula PnL não realizado.   
- v\_positions\_current: view consolidada de posições atuais.

# 3. Estrutura no Redis

### Chaves e TTLs:

- px:spot:<SYMBOL> → preço spot, TTL 5–10s  
- px:conv:USDTBRL → conversão BRL, TTL 30–60s  
- meta:symbols → lista de símbolos Binance, TTL 6–12h  
- news:top → últimas notícias RSS, TTL 120–300s

### Estratégia:

- Padrão cache-aside: consultar Redis → se não existir, buscar API → atualizar Redis.   
- TTLs ajustados para equilibrar frescor dos dados e economia de chamadas de API.   
- Uso preferencial de Redis para preços e notícias, reduzindo custo e latência.

# 4. Conclusão

Com esta modelagem, o Supabase/Postgres armazena dados persistentes e históricos, enquanto o Redis atua como cache de alta performance para dados temporais (preços e notícias). Essa combinação garante equilíbrio entre confiabilidade, velocidade e custo.