Fin-e Trip: Your Journey into Context-Aware FinCommerce using Vector Memory with Qdrant

January 22, 2026

Project Title & Overview

We built a **Context-Aware FinCommerce Engine** that blends **semantic understanding**, **financial reality**, **collaborative filtering**, and **real-time popularity tracking**. Qdrant powers the entire vector-native memory architecture with 4 specialized collections, enabling sub-second hybrid search with rich payload filtering.

In a sentence: embed everything (products, users, finances, interactions), retrieve semantically relevant options, then **re-rank by affordability, preferences, collaborative signals, and trending popularity**—all in real-time with full explainability.

Problem Statement why now?

E-commerce platforms excel at finding "relevant" items but fail to answer: *Can this person afford it? Do they trust this brand? What are similar users buying right now?* Traditional systems separate search, personalization, and trending—forcing users through disconnected experiences. We unify these with a **real-time vector-native architecture** that simultaneously reasons over semantic intent, financial constraints, user preferences, collaborative signals, and trending popularity—all in a single sub-second query.

Use Case Solved context-aware fincommerce

Scenario: "Laptop for machine learning under 1500." The system returns items that:

- Match semantic intent (specs, use-case, domain language)
- Respect personal taste (preferred brands like Apple, categories like Electronics)
- Stay within real-time affordability (balance + credit limit)
- Boost products similar users purchased (collaborative filtering with 7-day decay)
- Surface trending items (6-hour time-decayed popularity)
- Provide multi-reason explanations (" Trending", " Within budget", " Your preferred brand")

Real-time tracking: Every view, click, add-to-cart, and purchase is logged with weighted scores (0.1, 0.3, 0.6, 1.0) for immediate personalization.

How We Use Qdrant vector-native core

Qdrant is the **central vector memory and retrieval engine**, powering hybrid search (vector + payload filters) with low latency.

Collections & Payloads

Collection	Vector Dim	Distance	Key Payload Fields
`products_multimodal`	384	Cosine	product_id, name, categories[], brand, price, in_stock, image
`user_profiles`	384	Cosine	user_id, location, risk_tolerance, preferred_categories[], prefer
`financial_contexts`	256	Cosine	user_id, available_balance, credit_limit, current_debt, eligible
`interaction_memory`	384	Cosine	user_id, product_id, interaction_type, timestamp, weight, cat

Embedding Model: `all-MiniLM-L6-v2` (384D) with GPU acceleration (CUDA if available)
Payload Indexes: All collections use keyword indexes for efficient filtering by user_id, product_id, brand, categories

Query-Time Workflow

1. **Embed** user query → 384D vector using SentenceTransformer (GPU-accelerated)
2. **Vector Search** on `products_multimodal` with payload filters (price max_price)
3. **Retrieve Context** from `user_profiles` + `financial_contexts` by user_id
4. **Collaborative Filtering**: Compute user behavior vector from last 10 interactions (7-day decay)
5. **Find Similar Users**: Search `interaction_memory` excluding self, extract collaborative scores
6. **Popularity Scores**: Aggregate interactions in last 24h with 6-hour time decay, normalize
7. **Rerank**: Combine 5 signals (semantic, affordability, preference, collaborative, popularity)
8. **Explain**: Generate multi-reason explanations for each result
9. **Log View**: Auto-track all displayed products as "view" interactions (weight=0.1)

Performance: Typical query completes in ˜800ms (embedding: 50ms, search: 200ms, rerank: 150ms, popularity: 200ms)

Reranking Formula (5-Signal Hybrid)

$$\text{final\_score} = 0.30\, s_{\text{sem}} + 0.25\, s_{\text{aff}} + 0.15\, s_{\text{pref}} + 0.20\, s_{\text{collab}} + 0.10\, s_{\text{pop}}$$

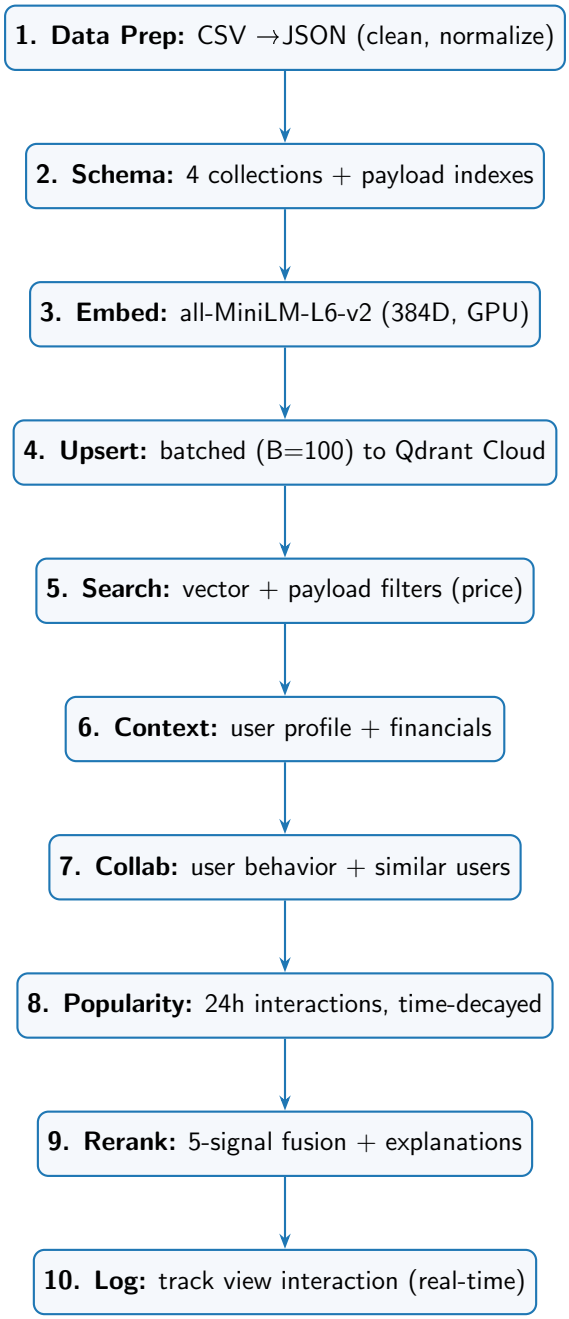
Where:

$$
\begin{aligned}
s_{\text{sem}} &= \text{cosine similarity from Qdrant search} \\
s_{\text{aff}} &= \max\left(0, 1 - \frac{\text{price}}{\text{balance} + \text{credit\_limit}}\right) \\
s_{\text{pref}} &= \max\left(\frac{|\text{user\_brands} \cap \text{product\_brand}|}{|\text{user\_brands}|}, \frac{|\text{user\_cats} \cap \text{product\_cats}|}{|\text{user\_cats}|}\right) \\
s_{\text{collab}} &= \frac{\sum_{i \in \text{similar users}} w_i \cdot \text{similarity}_i}{\max(\text{all collaborative scores})} \\
s_{\text{pop}} &= \frac{\sum_{j \in \text{interactions}} \text{weight}_j \cdot e^{-\lambda \Delta t_j}}{\max(\text{all popularity scores})}
\end{aligned}
$$

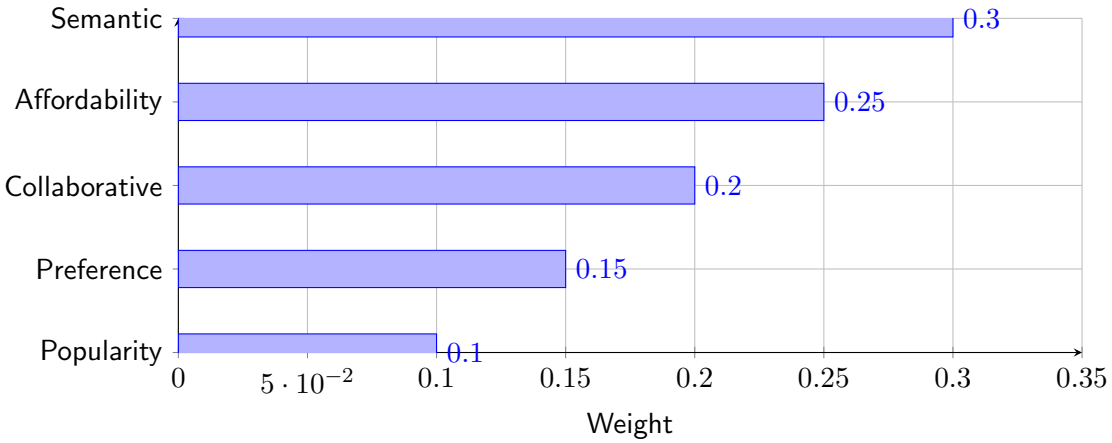
Time Decay Constants:

- Collaborative: $\lambda_{\text{collab}} = \ln(2)/(7 \times 86400)$ (7-day half-life)
- Popularity: $\lambda_{\text{pop}} = \ln(2)/(6 \times 3600)$ (6-hour half-life)

Architecture at a Glance complete flow



Score Weights (Visual) 5-signal blend



Implementation Highlights production-ready

Real-Time Interaction Tracking

File: `interaction_logger.py`

- 4 interaction types with weighted scores: view (0.1), click (0.3), add_to_cart (0.6), purchase (1.0)
- Generates 384D behavioral embeddings: "user purchased MacBook Pro price 1299 for laptop ML"
- Real-time upsert to `interaction_memory` collection
- Auto-triggered on product views, clicks, cart additions, purchases in Streamlit UI

Trending Products

Function: `get_top_interacted_products()`

- Aggregates interactions by product_id in last 24 hours (configurable)
- Exponential time decay: $w(t) = w_{\text{base}} \cdot e^{-\lambda(t_{\text{now}} - t_{\text{interaction}})}$
- Normalized popularity scores (0-1 range)
- Cold-start friendly: new products default to 0.0 (no penalty)

Collaborative Filtering

Method: User-based CF via interaction vector similarity

- Construct aggregate user behavior vector from last 10 interactions (7-day decay)
- Search for similar users' interactions (exclude self)
- Weight by (similarity $\times$ interaction_weight)
- Boost products that similar users clicked/purchased

Multi-Reason Explanations

Each product includes structured explanations across 5 categories:

- **Popularity:** "Very popular in last 24 hours", "Getting attention"
- **Affordability:** "Well within your budget", "Stretches your budget slightly"
- **Collaborative:** "Similar users purchased this", "Similar users viewed this"
- **Preference:** "Matches your preferred brand: Apple", "In your favorite category: Electronics"
- **Semantic:** "Strong match to your search query", "Moderately relevant"

Technology Stack

- **Vector DB:** Qdrant Cloud (4 collections, 384D/256D vectors)
- **Embeddings:** SentenceTransformer (all-MiniLM-L6-v2, GPU-accelerated)
- **Backend:** Python 3.x with type hints, comprehensive logging
- **Frontend:** Streamlit with real-time interaction hooks
- **Data:** 4 e-commerce datasets (Amazon, Walmart, Lazada, Shein)

Key Metrics

Metric	Performance
Query Latency	~800ms (end-to-end)
Embedding Speed	50ms (GPU), 150ms (CPU)
Interaction Log	¡50ms (single upsert)
Popularity Fetch	~200ms (1K interactions/24h)
Reranking	~150ms (10 products)
Vector Dimension	384 (products, users, interactions), 256 (financials)
Cold Start	Graceful (defaults to 0.0, no errors)

Conclusion

Our Context-Aware FinCommerce Engine demonstrates how vector databases like Qdrant enable **real-time personalization at scale**. By unifying semantic search, financial constraints, user preferences, collaborative signals, and trending popularity into a single sub-second query, we deliver hyper-personalized recommendations that are both relevant and realistic. The system is production-ready with comprehensive logging, multi-reason explanations, and graceful cold-start handling—ready to transform e-commerce experiences.