MERCADO AI

The Agentic Automated Rancher Exchange

An AI-Native Operating Network for Livestock Markets, Logistics, Traceability, Risk, and Settlement

White Paper - 2026 Architecture and Product Update

Author: Cyril Simone

September 2026

Executive Summary

Mercado AI is evolving from a collection of marketplace dashboards into an agentic operating network for small and medium U.S. cattle ranches. The updated Automated Rancher Exchange coordinates livestock inventory, market discovery, auctions, pooled transportation, processor and feedlot capacity, animal health signals, ownership/traceability, financial performance, and settlement through a common event-driven data model.

The defining product capability is the Load Pool and Transport Quote Snapshot. Rather than showing a static freight estimate, Mercado AI continuously recalculates the expected cost of moving a rancher's animals as nearby compatible ranches enter or leave a shared load, route constraints change, fuel assumptions move, or processor destinations change. Every price change is versioned, traceable, explainable, and classified as estimated, exact, or locked.

Six specialized agents sit above deterministic business services. The agents explain, recommend, and orchestrate; they do not invent prices, routes, ownership, health status, or settlement values. This separation is central to the 2026 architecture: probabilistic AI handles language, discovery, reasoning, and user interaction, while deterministic services remain authoritative for money, routing constraints, auctions, compliance, and blockchain commitments.

1. Why the Architecture Changes Now

Agentic AI has matured beyond simple chat interfaces. Modern agent frameworks now support structured tools, externalized state, sandboxed execution, handoffs, durable workflows, and Model Context Protocol (MCP)-style integrations. Mercado AI can use these capabilities without turning the livestock exchange into an uncontrolled multi-agent system. The recommended pattern is a Rancher Concierge manager that delegates to five specialists and invokes deterministic tools with explicit authorization boundaries.

The data layer is also more capable. PostgreSQL can serve as the transactional system of record while pgvector adds semantic retrieval. Neo4j can combine graph relationships, vector retrieval, and GraphRAG-style context for explainable matching across ranches, lots, trucks, processors, feedlots, buyers, suppliers, and historical outcomes. This allows Mercado AI to reason about both similarity and network relationships while keeping financial truth in the relational ledger.

2. The Six-Agent Operating Model

Agent	Primary Role	Authoritative Tools	Key Outputs
Rancher Concierge	Single conversational entry point; intent, explanation, approvals, handoffs	Directory search, lot service, quote service, auction service, policy retrieval	Recommendations, explanations, action plans, approvals
Transport Cost Agent	Maintains running transport cost and cost-per-head as pools/routes change	Load Pool Service, Quote Engine, Allocation Engine, fuel/toll service	Estimated/exact/locked quote snapshots, delta explanations
Route & Pooling Agent	Groups compatible	Constraint solver, route	Candidate pools,

	ranch loads and optimizes multi-stop movement	matrix, capacity/welfare rules	routes, pickup sequence, savings
Auction & Market Agent	Manages listings, bids, market context, price signals	Auction engine, market feeds, buyer matching	Bid events, reserve guidance, market alerts
Health & Risk Agent	Surfaces health, movement, weather and operational risk	Animal records, sensor/event feeds, weather/risk rules	Alerts, risk scores, escalation recommendations
Financial & Settlement Agent	Tracks margin, fees, transport allocation, invoices and settlement	Ledger, payment service, blockchain commitment service	Per-head economics, settlement state, audit trail

3. Core Product Workflow

- Rancher creates or imports an Animal Lot with species/class, head count, weights, location, readiness window, health/traceability attributes, and destination preferences.
- The matching layer identifies compatible processors, feedlots, buyers, transporters, and nearby ranches using deterministic filters plus graph/vector ranking.
- A Load Pool is created for compatible shipments. The Route & Pooling Agent proposes pickup sequences and vehicle utilization scenarios.
- The Transport Cost Agent produces Transport Quote Snapshots. Each snapshot preserves assumptions, route version, pool membership, cost allocation method, confidence/state, and the exact reason for any change.
- The rancher can accept, reject, wait for more pool participation, choose another processor, or lock a quote. UI changes are optimistic but reversible until the authoritative service confirms them.
- Dispatch converts the accepted plan into operational commitments. GPS/movement events update the live dashboard and trigger exceptions.
- Final settlement reconciles actual mileage, wait time, fees, auction proceeds, processor charges, and contractual adjustments. Only after business records stabilize is the final commitment written to the blockchain audit layer.

4. Updated Data Architecture

PostgreSQL is the authoritative transactional store. Neo4j is the relationship and matching graph. pgvector provides semantic similarity for documents, participant profiles, requirements, and historical cases. Object storage holds certificates, inspection documents, images, and signed artifacts. A streaming/event layer distributes state changes to the dashboard and agents. Blockchain is an audit and ownership-commitment layer, not the primary operational database.

4.1 New and Revised PostgreSQL Tables

Table	Key Fields	Purpose
organization	id, type, legal_name, display_name, verification_status, tax_region, created_at	Canonical participant record for ranch, processor, feedlot, transporter, buyer, supplier, labor provider.

ranch_profile	organization_id, operation_scale, herd_size, production_model, geo_point, service_radius	Extends organization for ranch-specific attributes.
animal	id, current_owner_org_id, eid, species, breed, sex, birth_date, status	Individual animal identity where individual tracking is required.
animal_lot	id, owner_org_id, lot_class, head_count, avg_weight, readiness_start/end, origin_geo, status	Primary commercial grouping; replaces ambiguous listing/shipment naming.
animal_event	id, animal_id/lot_id, event_type, occurred_at, location, source, payload_json	Immutable operational timeline for health, movement, custody and status events.
load_pool	id, status, destination_type/id, pickup_window, target_capacity, current_capacity, solver_version	Central shared-transport aggregation object.
load_pool_member	id, load_pool_id, lot_id, ranch_org_id, head_count, allocated_weight, join_status	Membership and allocation within a pool.
route_plan	id, load_pool_id, version, vehicle_id, total_miles, duration_minutes, utilization_pct, constraint_score	Versioned deterministic route output.
route_stop	id, route_plan_id, sequence, stop_type, organization_id, geo_point, eta, service_minutes	Farm/feedlot/processor stops.
transport_quote_snapshot	id, load_pool_id, route_plan_id, quote_version, state, total_cost, ranch_cost, cost_per_head, valid_until, trace_id	Versioned running cost record. State = ESTIMATED, EXACT, LOCKED, EXPIRED, SETTLED.
transport_cost_component	id, quote_snapshot_id, type, quantity, rate, amount, allocation_method	Fuel, mileage, driver, toll, wait, cleaning, insurance, platform fee, etc.
quote_change_explanation	id, quote_snapshot_id, previous_snapshot_id, reason_code, delta_amount, explanation_text	Explanation-first UX and auditability.
shipment	id, lot_id, accepted_quote_id, origin_id, destination_id, status, planned_departure/arrival	Commercial movement commitment.
dispatch	id, shipment_id, transporter_org_id, vehicle_id, driver_id, status, actual_departure/arrival	Operational execution of shipment.
auction	id, lot_id, seller_org_id, reserve_price, starts_at, ends_at, status	Auction header.

bid	id, auction_id, bidder_org_id, amount, placed_at, status, trace_id	Append-only bid record.
market_price_snapshot	id, commodity_class, region, price, unit, source, observed_at	Market context; separate from auction truth.
health_observation	id, animal_id/lot_id, observation_type, value, unit, severity, source, observed_at	Health/sensor/manual observations.
risk_alert	id, entity_type/id, risk_type, severity, status, detected_at, rule_or_model_version, explanation	Operational and health risk alerts.
financial_ledger_entry	id, organization_id, lot_id, shipment_id, entry_type, debit, credit, currency, occurred_at	Double-entry-ready financial truth.
settlement	id, transaction_id, seller_org_id, buyer_org_id, gross_amount, deductions, net_amount, status	Commercial settlement.
ownership_event	id, animal_id/lot_id, from_org_id, to_org_id, event_type, effective_at, transaction_id, chain_commitment_id	Custody/ownership history.
blockchain_commitment	id, entity_type/id, network, contract_address, tx_hash, content_hash, committed_at, status	Stores blockchain proof without making chain the system of record.
agent_run	id, agent_type, user_id, trace_id, started_at, ended_at, status, model, policy_version	Agent observability.
agent_tool_call	id, agent_run_id, tool_name, input_hash, output_hash, authorization_state, latency_ms, status	Tool audit and safety.
event_outbox	id, aggregate_type/id, event_name, payload_json, trace_id, created_at, published_at	Transactional outbox for reliable event delivery.
consent_grant	id, organization_id, data_category, purpose, grantee_type/id, scope, starts_at, ends_at, status	Controls directory/data monetization and sharing.
data_product_subscription	id, subscriber_org_id, product_id, tier, usage_limit, revenue_share_rule_id, status	Supports monetized directory/data products.
revenue_share_rule	id, rule_name, contributor_type, basis, percentage, effective_from/to	Defines contributor compensation for eligible data products.

4.2 Neo4j Graph Model

The graph should mirror identities from PostgreSQL by stable UUID rather than become a second system of record. Recommended node labels include Ranch, Processor, Feedlot, Transporter, Buyer, Supplier, LaborProvider, AnimalLot, LoadPool, Vehicle, Auction, Market, Region, Certification, and DataProduct.

High-value relationships include OWNS, LISTS, ACCEPTS_CLASS, CAN_SERVE, NEAR, PARTICIPATES_IN, POOLS_WITH, HAULS_TO, BID_ON, PURCHASED, PROCESSED_AT, FEEDS_AT, CERTIFIED_AS, SUPPLIES, WORKED_WITH, PREFERS, and CONTRIBUTES_DATA_TO. Embeddings should be attached to profiles, requirements, lots, service descriptions, and historical outcomes for semantic matching. Graph traversal then constrains or explains why a semantic match is operationally relevant.

5. Agentic Technology Stack - 2026 Update

- FastAPI + Pydantic v2 for typed service contracts and agent tools.
- PostgreSQL 18.x-class deployment for transactional data, with a current pgvector release for vector similarity. Keep vector indexes patched because pgvector has had security-sensitive fixes in 2026.
- Neo4j 2026.01+ where practical to use newer vector SEARCH/in-index filtering capabilities; use the maintained neo4j-graphrag Python package rather than deprecated neo4j-genai.
- Agent orchestration through a manager/specialist pattern using a modern Agents SDK or equivalent framework with structured tool calling, durable state/checkpoints, explicit permissions, and MCP-compatible integrations.
- WebSocket/SSE event delivery for quote changes, auction bids, tracking, alerts, and agent status. Use an event broker such as Kafka, Redpanda, or managed cloud equivalents as scale requires.
- OR-Tools or equivalent deterministic vehicle-routing solver as the baseline; optionally layer learned ranking or forecasting around it, never in place of hard animal-welfare, capacity, time-window, and legal constraints.
- React/Next.js authenticated dashboard with a shared design system, typed API client, optimistic/reversible mutations, and explicit ESTIMATED / EXACT / LOCKED cost badges.
- Object storage for documents and media; GraphRAG retrieval for regulations, processor requirements, operating procedures, contracts, and ranch-specific knowledge.
- Polygon/EVM-compatible smart contracts or another suitable low-cost chain for ownership/settlement commitments after dispatch and settlement records are finalized.
- OpenTelemetry-compatible traces across API, agent, solver, auction, and settlement flows. Every quote and dispatch action carries a trace_id.

6. Running Transport Cost: The Product's Economic Core

The Transport Cost Agent should never calculate freight by free-form language-model reasoning. It calls deterministic services and explains the returned result. A quote snapshot is recalculated whenever pool membership, route, vehicle, rates, destination, timing, or constraints change.

A simplified allocation model is: total route cost = fixed dispatch cost + mileage cost + fuel surcharge + driver/time cost + tolls + stop/service cost + livestock-specific handling + risk/insurance + platform fee. Each ranch's allocation can be computed using a transparent policy such as marginal-mile contribution,

weight/head share, occupied trailer capacity, stop burden, or a hybrid. The policy and version are stored on every snapshot.

Example: Ranch A initially requires a dedicated trip at $1,080 for 18 head ($60/head). Ranch B joins a compatible pool and increases the route by only 34 marginal miles while improving trailer utilization. The new pooled total might be $1,420, with Ranch A allocated $760 ($42.22/head) and Ranch B allocated the remainder under the configured allocation policy. The dashboard must show the $17.78/head reduction and explain that the change came from higher utilization and shared fixed/mileage costs. These figures are illustrative, not production pricing.

7. Real-Time Dashboard

- Live Animal Tracking: lot/animal status cards and map timeline across Farm -> Feedlot -> Transport -> Processor, with custody and ownership separated.
- Transport Cost Ribbon: always-visible current cost, cost/head, quote state, savings from pooling, pool fill percentage, quote expiration, and last recalculation reason.
- Route Optimization: route map, pickup sequence, utilization, miles, ETA, constraints, alternative processors, and 'wait for another ranch' scenario.
- Auction & Market: live bids, reserve status, market-price context, buyer matches, auction clock, and volatility alerts.
- Health & Risk: RFID/health observations, movement exceptions, weather/heat risk, documentation gaps, and human escalation.
- Financial Performance: expected sale proceeds, transport, feedlot/processor fees, platform fees, margin/head, realized settlement, and variance from estimate.
- Agent Activity Overlay: six agents with status = IDLE, ANALYZING, ACTION_REQUIRED, LOCKED, ALERT, or DEGRADED; users can inspect what tool/data caused each recommendation.

8. Event Naming and API Discipline

Use one event grammar: <domain>.<entity>.<past-tense-action>.v1. Examples: transport.load_pool.member_joined.v1, transport.quote.recalculated.v1, transport.quote.locked.v1, routing.route_plan.optimized.v1, auction.bid.placed.v1, tracking.lot.arrived.v1, health.risk_alert.created.v1, finance.settlement.completed.v1, ownership.transfer.committed.v1.

All externally meaningful mutations return trace_id, entity version, authoritative status, and occurred_at. Agent responses should cite the tool result internally and display the user-facing explanation without exposing chain-of-thought.

9. Directory and Data-Network Monetization

The expanded Mercado Directory becomes a permissioned data network rather than a static listing. Ranchers, processors, feedlots, transporters, veterinarians, feed/forage suppliers, labor providers, grazing-lease providers, insurers, lenders, auction markets, equipment/service providers, and verified buyers can participate. The directory feeds matching and logistics while also creating potential data products.

Revenue should be tied to consent and value creation. Examples include premium directory placement, verified lead generation, transaction fees, pooled-logistics savings share, anonymized/aggregated market intelligence, processor-capacity intelligence, and API subscriptions. Raw identifiable ranch data should not be resold by default. The consent_grant, data_product_subscription, and revenue_share_rule tables make contributor rights and compensation explicit.

10. Safety, Governance, and Human Control

- Agents recommend and orchestrate; deterministic services authorize money, bids, route constraints, ownership changes, and settlement.
- Require explicit user confirmation for bid placement, quote locking, dispatch commitment, payment, and ownership transfer.
- Apply least-privilege tool permissions and organization-level row security.
- Log every tool call and major decision with trace IDs; retain quote snapshots rather than overwriting history.
- Use retrieval grounding for regulations and processor requirements; label model-generated interpretation as guidance rather than legal/veterinary advice.
- Implement agent circuit breakers, rate limits, degraded-mode behavior, and manual override paths.
- Separate custody, legal ownership, physical location, and commercial title because they can change at different times.

11. Phased Implementation

Phase	Primary Deliverable
Phase 0 - Canonical Model	Shared types, PostgreSQL schema, IDs, event grammar, API contracts, migrations.
Phase 1 - Economic Core	Animal Lot, Load Pool, Route Plan, Quote Snapshot, allocation engine, Transport Cost Agent.
Phase 2 - Concierge + Directory	Rancher Concierge, graph directory, vector/GraphRAG retrieval, processor/feedlot/transporter matching.
Phase 3 - Real-Time Operations	WebSocket/event layer, live dashboard, GPS/tracking, auction events, health/risk alerts.
Phase 4 - Financial Closure	Ledger, invoicing, settlement, margin analytics, reconciliation.
Phase 5 - Blockchain Proof	Ownership/custody commitments and final settlement hashes after operational model stabilizes.
Phase 6 - Data Network	Consent, premium directory, aggregated data products, APIs, revenue-sharing rules.

12. What Changes From the Earlier Mercado AI Prototype

- Replace separate feature-centric dashboards with a shared agentic operating core and one live dashboard.

- Make Animal Lot, Load Pool, Transport Quote Snapshot, Shipment, Dispatch, Ownership Event, and Settlement canonical entities.
- Move transport economics from a static estimate to a versioned, continuously recalculated product primitive.
- Use AI for conversation, retrieval, ranking, explanation, and orchestration; use deterministic code for authoritative calculations and commitments.
- Treat Neo4j as a relationship/matching layer and PostgreSQL as transactional truth rather than duplicating mutable truth across databases.
- Add consent and revenue-sharing structures so the participant directory can become a defensible, permissioned data network.
- Delay blockchain finalization until quote, dispatch, ownership, and settlement semantics are stable.

13. Strategic Outcome

The updated Mercado AI is not simply a digital cattle marketplace. It is an operating network that makes fragmented livestock supply-chain capacity computable. Its defensibility comes from the combination of participant density, historical transaction and route outcomes, graph relationships, continuously improving matching, transparent pooled-logistics economics, trusted traceability, and a permissioned data network.

For small and medium ranches, the immediate value proposition is measurable: more buyer and processor options, lower transportation cost through pooling, faster visibility into processing capacity, better market timing, auditable animal movement and ownership, and a single conversational interface that explains the economics before the rancher commits.

References and Technology Notes

1. OpenAI. “The next evolution of the Agents SDK.” April 15, 2026. https://openai.com/index/the-next-evolution-of-the-agents-sdk/
2. Neo4j. GraphRAG for Python documentation, current 2026 release. https://neo4j.com/docs/neo4j-graphrag-python/current/
3. Neo4j. GraphRAG/vector retrieval documentation, including Neo4j 2026.01 SEARCH and in-index filtering. https://neo4j.com/docs/neo4j-graphrag-python/current/user_guide_rag.html
4. PostgreSQL Global Development Group / pgvector. pgvector release information, 2026. https://www.postgresql.org/about/news/pgvector-082-released-3245/
5. USDA APHIS. Animal disease traceability and electronic identification requirements should be treated as external regulatory sources and integrated through versioned compliance retrieval rather than hard-coded agent knowledge.

About the Author

Cyril Simone is a technology executive and computer scientist focused on artificial intelligence, data science, blockchain, quantum computing, and the application of emerging technology to business transformation. As CTO of Mercado AI, his work centers on using AI-driven markets, logistics

optimization, graph intelligence, agentic systems, and digital traceability to improve the economics and resilience of agricultural supply chains.