Scalable Sales GenAI Architecture

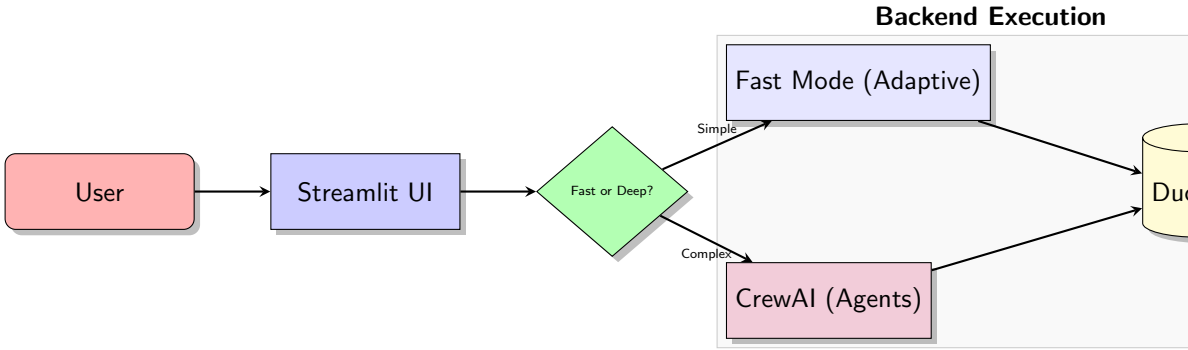
Production-Ready CrewAI + OpenAI + DuckDB

Abhishek Dangwal

February 8, 2026

Architecture Overview for 100GB Scale Implementation

High-Level System Architecture



Key Components:

- **UI:** Streamlit with structured rendering (SQL + Data + Insights).
- **Orchestration:** CrewAI for agent delegation; Fast path for low latency.
- **Data:** Persistent DuckDB using `.cache/` storage.

LLM Integration Strategy

Fast Mode (Low Latency)

- **Goal:** Speed & Low Cost.
- **Technique:** Adaptive Schema-Driven SQL Synthesis.
- **Model:** `gpt-4o-mini`.
- **Use Case:** Simple lookups, standard aggregations.
- *Bypasses heavy agent loops.*

Deep Mode (High Fidelity)

- **Goal:** Complex Reasoning.
- **Technique:** CrewAI Multi-Agent System.
- **Workflow:**
  - Schema Chunking (Reduce context).
  - SQL Validation Agent.
  - Auto-Repair loop.

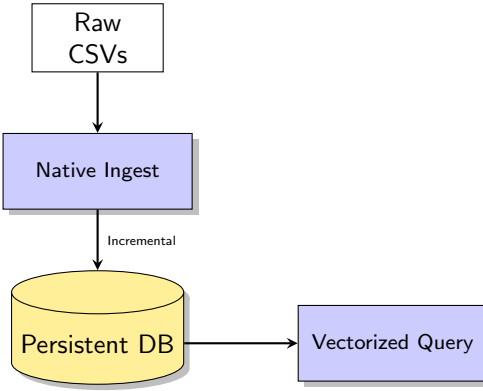
Data Storage & Retrieval: 100GB Scale Design

DuckDB Architecture Strategy:

- **Persistent Storage:** Data stored in `.cache/sales.duckdb` to prevent full reloads.
- **Native Ingestion:** Uses `read_csv_auto` (Zero-copy) instead of Pandas loading for lower memory footprint.
- **Dynamic Marts:** Auto-generation of `mart_dyn_*` views for new CSV files.

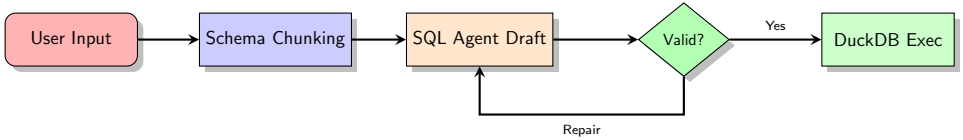
Performance at Scale:

- Vectorized Execution Engine (OLAP optimized).
- Incremental refresh checks (Time-stamp based).



Example Query-Response Pipeline

Scenario: "Show top 5 regions by sales in Q3."



Pipeline Steps

1. **Schema Chunking:** Retrieve only relevant table definitions (Cost saving).
2. **Drafting:** Agent generates SQL grounded in Dataset Playbook.
3. **Guardrails:** Validator checks for `SELECT`-only safety and column existence.
4. **Visuals:** Streamlit renders SQL code block + DataFrame results.

Cost and Performance Considerations

Optimization Area	Implementation Strategy
Token Cost	Schema Chunking (Only send relevant headers) Use of `gpt-4o-mini` for 80% of queries
Latency	**Response Caching** for repeated queries Fast Mode for non-reasoning tasks
Ingestion Speed	`read_csv_auto` (DuckDB) vs Pandas Incremental file loading (Timestamp checks)

Future Improvements:

- Support for Parquet (Compression for >100GB).
- Async background workers for request queuing.

Thank You

Questions? email me at:dangwalabhishek5@gmail.com