Extending the OMOP Lifecycle

How AI Helped Us Revive Our OMOP

Roger Carlson
Corewell Health | SASS Honest Broker Analytics
ACQIRE: Ask Clinical Questions In Regular English

Where the OMOP Pipeline Traditionally Ends

```
Source (Epic) → ETL → CDM tables → DQD → Achilles → [end]
```

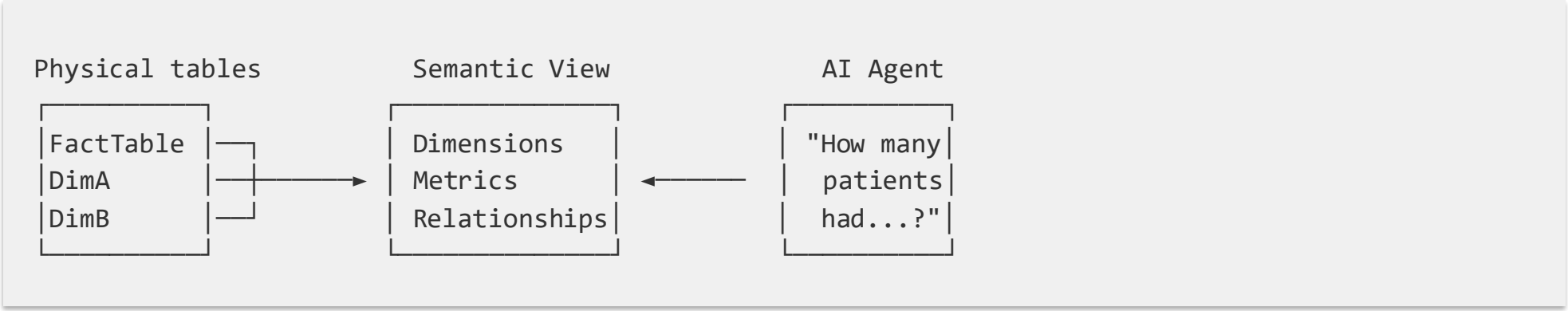
- DQD checks structure. Achilles checks distributions.
- Neither checks whether the CDM can reproduce the results the source data produced.
- We assumed our CDM was correct because the pipeline ran cleanly.

We were wrong by about 90.4 million rows.

What Is a Snowflake Semantic View?

The object that makes source tables queryable in natural language

- A **Semantic View** is a Snowflake object that defines a business-level data model: one fact table, joined dimensions, metrics, and filters — all declared in DDL
- It sits on top of physical tables but exposes them as named concepts (dimensions, measures) rather than raw columns and joins
- A Cortex Agent can query a semantic view in natural language — the platform translates English questions into correct SQL using the declared relationships
- No data is copied or materialized — the semantic view is a contract describing how tables relate, not a pipeline
- This matters for OMOP: each PULL_ query already encodes the same structure (grain, join paths, filters, output columns). The shape is identical.



PULL_ Queries Are Already Semantic View Blueprints

- Our dbt pipeline: source_stg → PULL_ → STAGE_ → WRITE_RAW → CDM
- Each PULL_ query encodes validated join paths, required filters, SCD2 dedup, vocabulary resolution
- A PULL_ query and a Snowflake Semantic View have the same shape:
  - One fact table, joined dimensions, required filters, defined grain
- **The PULL_ query already solved the join path. We just exposed it to an AI.**

- Cortex Code is Snowflake's implementation of Claude Code — an AI coding agent with full access to the warehouse, schema metadata, and SQL execution.
- We used Cortex Code to
  - read our existing PULL_ models
  - identify the structural equivalence
  - generate the semantic view DDL

The Clarity Experiment and the Caboodle Pivot

Cortex Code built 15 Clarity semantic views from our existing PULL_ layer.

But we wanted to compare to semantic views based on Caboodle queries, which most of our Honest Brokers use.

We used Cortex Code to convert our entire PULL_ layer from Clarity-sourced to Caboodle-sourced (1 week)

- Read existing Clarity PULL_ models and Caboodle schemas simultaneously
- Generated equivalent Caboodle PULL_ models preserving the same output contract
- PULL_ queries dropped from 60-80 lines to 15-25 lines — fewer joins, fewer silent failures
- Vocabulary resolution simplified: Caboodle stores decoded values alongside codes, making STCM mapping transparent

Caboodle demonstrated its superiority empirically at every step: simpler queries, more complete results, fewer points of failure

Rule adopted: Caboodle first. Clarity only when Caboodle lacks the table.

Published on Epic User Web: [OMOP VI: The Undiscovered Caboodle (userweb.epic.com/Thread/147477)](userweb.epic.com/Thread/147477)

Semantic Views for Epic Caboodle: OMOP-Aligned Clinical Data Access

Real Honest Broker Queries Against the SVs

The semantic views are built from OMOP join patterns, but they query the source Caboodle and Clarity tables directly — not the CDM.

- A Cortex Agent backed by these semantic views became operational: analyst submits a clinical question in English, receives validated SQL
  - Validation methodology: AI never sees human SQL — only a clinical specification document
  - 12 validation batches, 105 projects evaluated:
  - **76/76 validatable = 100%**
- Cortex Code performed each validation: generating queries from clinical specs, running patient-level comparisons against historical analyst results, surfacing discrepancies
- Every validated project surfaced institutional gotchas → ~500 knowledge patterns fed back

But this knowledge was not being fed back into OMOP. The source-table SVs were getting smarter while the CDM stayed static. They were diverging.

What the Queries Found Wrong with OMOP

Some domains: 100% from source, 0% from CDM. Discovered unknown cross-domain vocabulary routing.

Gap	Dropped Rows	Affected Patients
ICD-10 Condition → Observation	9,600,003	1,304,062
SNOMED Observation → Procedure	68,518,813	2,772,317
SNOMED Observation → Measurement	5,472,630	973,623
ICD-9 Condition → Observation	4,283,892	477,326
ICD-10 Condition → Measurement	2,548,590	103,039
Total	**~90.4M**	**~2.8M unique**

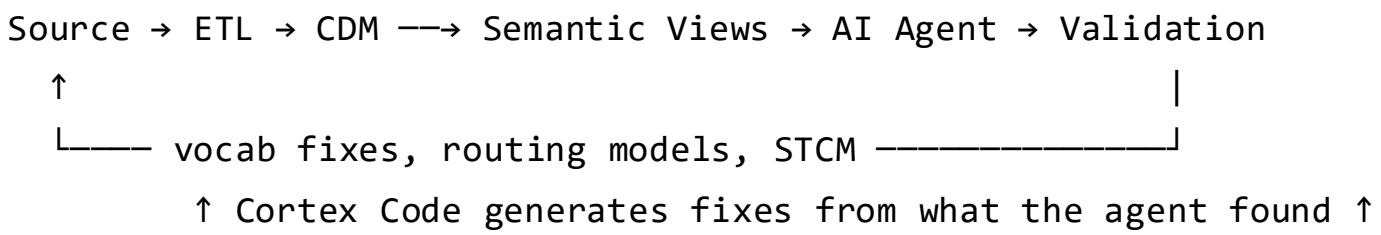
No DQD violation. No Achilles anomaly. Patient-level comparison found what standard tools could not.

Feeding Results Back: The Loop Closes

Semantic view queries discover defects. Defects become new PULL_ models. OMOP gets richer.

- Cortex Code traced the discrepancies upstream: patient-level diffs revealed ICD codes crossing vocabulary domains, silently dropped by INNER JOINs
- Built 6 cross-domain routing models — Cortex Code generated each from the pattern it discovered during validation
- Rebuilt the entire PULL_ layer as Caboodle-first with Cortex Code (60-80 line queries → 15-25 lines), validated model-by-model against Clarity for patient-level equivalence
- STCM improvements driven by validation gaps: ~50 mapping files created or corrected

Extended Lifecycle:



What Flowed Back to OMOP

What Changed	Measure
Cross-domain rows recovered	90.4M (2.8M patients)
New routing models built	6 (ICD-10, ICD-9, SNOMED cross-domain)
PULL_ layer converted	Clarity → Caboodle (60-80 lines → 15-25)
STCM mapping files created/corrected	~50
Domains added to CDM coverage	Previously 0% from OMOP, now populated

Cortex Code's role at each stage:

- **Construction**: built semantic views from PULL_ patterns
- **Testing**: validated queries against historical analyst results
- **Diagnosis**: traced discrepancies to upstream ETL defects
- **Repair**: generated new PULL_ models and routing fixes

What's Next: Semantic Views Generate New PULL_ Queries

We proved this with the 6 cross-domain routing models — each was generated from a pattern Cortex Code discovered when an SV returned results that OMOP could not.

The next steps:

1. Every SV encodes a validated join path to source data
2. If that SV returns results but the corresponding OMOP domain is empty or undercounted, the gap is identified
3. The SV's structure (fact table, dimensions, filters, grain) is already the blueprint for a new PULL_ model
4. Cortex Code generates the PULL_ query directly from the SV definition — the CDM becomes self-healing

Just as we used OMOP to create the original Semantic Views, now we use Semantic Views to improve OMOP.

The CDM improves when you use it hard enough to find its failures.

Roger Carlson | Corewell Health
SASS Honest Broker Analytics
ACQIRE: Ask Clinical Questions In Regular English