Vamshi Jandhyala

The catalog becomes the query interface

✠

September 2023

An essay from 2023 arguing that natural-language queries would replace SQL as the primary interface to enterprise data, with the data catalog as the substrate that makes those queries trustworthy. Republished with a 2026 note.

Editor's note, April 2026

I wrote this in mid-2023, when most enterprise data catalogs were still being sold as governance tools and the canonical interface to data was a SQL editor. The piece argued the opposite: that the catalog would become the *query* interface, and that the language of that interface would be natural language, not SQL. By 2026 that's the default vendor pitch. Snowflake Cortex, Databricks Genie, and every catalog vendor's 2025 release positions a natural-language query layer as the headline feature, with the catalog as the substrate that makes its answers trustworthy.

The argument is preserved with citations linked through to source.

The query is already in natural language

Look at how a financial analyst, a portfolio manager, or a risk manager actually asks for data. They do not write SQL. They ask:

- *Retrieve historical price data for Amazon for the past ten years.*
- *Get real-time price data for all assets in Portfolio X.*
- *Show current credit ratings for all companies in Sector Y.*
- *Pull intraday price data for all tech stocks in the NASDAQ.*
- *Get a list of all transactions made by the company in the last month for compliance checks.*

Then somebody, somewhere, translates that into SQL or into a Bloomberg terminal command or into a workflow ticket to a data engineer. The translation step is friction. It costs days, sometimes weeks. It introduces errors. It makes the analyst dependent on the translator.

The query is already in natural language. The interface is what is wrong.

What the catalog gives the natural-language layer

The reason natural language has not replaced SQL until now is that natural language alone is not a query system. It is a request format. To resolve a request reliably, the system underneath needs to know what data exists, where it lives, what it means, who is allowed to see it, and how trustworthy it is. That substrate is exactly what a well-built data catalog provides.

A natural-language query interface that does not sit on top of a catalog is a chatbot. A natural-language query interface that does sit on top of a catalog is a query system. The difference is whether the answer can be trusted.

Six properties the catalog must provide for the natural-language layer above it to work:

1. **Semantic search.** The catalog must understand that a request for *"Q2 revenue"* maps to a particular dataset, table, and field across many possible naming conventions. Keyword search is not enough. The catalog needs to capture the meaning of a request, not just the spelling.

1. **Business glossary integration.** When the user asks for *churn*, the system must know how *churn* is defined inside this organisation. A glossary that lives next to the catalog gives the natural-language layer a vocabulary that matches the business, not a generic dictionary.

1. **Lineage and provenance.** When the layer answers, the user must be able to ask: where did this come from? Lineage from raw source through transformations to the answered field is the audit trail that makes a generated answer defensible. Without it the answer is plausible but not trustworthy.

1. **Data quality signals.** Two datasets that match the same query are not equally good. The catalog must carry quality metadata (freshness, completeness, certification, owner endorsement) and the natural-language layer must rank by it. The user wants the right answer, not any answer.

1. **Access control integration.** The layer must respect the same row-level and column-level permissions the rest of the data infrastructure does. A natural-language interface that sees more than the user is allowed to see is a leak. Enforcement happens at the catalog, not at the chat layer.

1. **Federation across catalogs.** Most large organisations run more than one catalog. The natural-language layer must be able to ask the same question across all of them and rank results consistently. Federation is what turns a single-source assistant into an enterprise interface.

These are the six. The 2023 industry was selling fifty more features as differentiators. Most were noise. These six are the load-bearing ones.

What the catalog gives the user directly

Independent of the natural-language layer, the catalog still owns the workflow that the layer must satisfy. The canonical access path inside an enterprise data marketplace has nine steps:

1. **Identify the need for data.** The analyst, scientist, or engineer recognises a question that requires data they do not yet have.
2. **Access the catalog.** A searchable interface presenting datasets, fields, owners, lineage, quality, and access status.
3. **Search for relevant assets.** Keywords, filters, semantic search, persona-aware recommendations.
4. **Evaluate.** Inspect metadata, sample rows, freshness, ownership, lineage.
5. **Request access.** Specify purpose, intended use, and duration where data is gated.
6. **Compliance check.** Privacy, regulation, and policy review.
7. **Receive access.** Through a database connection, an API, or an extraction process.
8. **Consume.** Query, download, integrate, or feed into a model.
9. **Feedback.** Rate the dataset, leave annotations, signal usefulness or quality issues to the next user.

A natural-language layer that intends to replace SQL must produce the same outcome that this nine-step path produces, with the same audit trail. It is not a faster way of running a query. It is a faster way of running the workflow.

Why this matters for a financial data marketplace

In a financial services context the personas with the strongest natural-language demand are also the ones least likely to write SQL.

- **Financial analysts** ask for historical prices, financial reports, ESG data, latest news, dividend histories. These map cleanly onto reference datasets.
- **Portfolio managers** ask for real-time prices, performance attribution, risk metrics, ESG scores, analyst ratings, all scoped to portfolios.
- **Risk managers** ask for credit ratings, volatility histories, liquidity metrics, default probabilities, sector-level risk factors.
- **Traders** ask for real-time quotes, intraday volumes, sector-wide tape, futures and options chains.
- **Compliance officers** ask for regulatory changes, transaction logs, AML lists, audit trails of who accessed what.

Every one of these queries is naturally phrased in business English, scoped to a portfolio or a sector or a time window, and constrained by access rules. SQL is the wrong native interface. The terminal is the wrong native interface. A catalog with a natural-language layer on top is the right native interface, because it can carry the scope, the access control, and the trust signals all the way through.

What this means for the people building data products

Three implications follow.

First, the catalog is no longer a governance tool. It is a query substrate. Build it as such. The investment in metadata, lineage, glossary, and quality is now the investment in the primary user interface, not in compliance overhead.

Second, the natural-language layer is competitive, not differentiating. Every vendor will ship one. The competitive surface is the quality of the catalog underneath, the breadth of the data it covers, and the trustworthiness of the answers it produces. The chat box is not the moat. The substrate is.

Third, the role of the data team shifts. Less time is spent translating questions into SQL. More time is spent curating the catalog: defining business terms, certifying datasets, tracing lineage, encoding access rules, signalling quality. The data team's output stops being queries and starts being a knowledge graph.

The catalog becomes the query interface. The query layer becomes natural language. The substrate is everything.