Claude + MCP to Snowflake vs. Snowflake Cortex

Same analytics powers, different place where the AI actually runs

Explainer · June 4, 2026

ABSTRACT

Most people assume this is a feature bake-off: which option lets you do natural-language querying, search, or retrieval over your data? On that axis, the two paths look nearly identical, because the external route can call the very same Snowflake services. The real difference is architectural, not feature-level. Snowflake Cortex runs AI *inside* Snowflake's perimeter, next to your data, while Claude connected through a Model Context Protocol (MCP) server is an external agent that reaches in, calling Cortex services as tools and additionally coordinating work across systems that have nothing to do with Snowflake. The practical implications: choose Cortex when your data and controls already live in Snowflake; add MCP when an external agent needs to front-end Snowflake, stitch in other tools, or operate under explicit tool-by-tool permissions. The two are complementary, not rival.

Keywords: Snowflake Cortex; Model Context Protocol; Claude; agentic AI; NL-to-SQL; retrieval-augmented generation; data governance; orchestration

1. Introduction

Anyone following enterprise AI has heard two stories at once. The first: data warehouses are growing AI features so you can "talk to your data" without writing SQL. The second: AI agents are learning to use tools, reaching out of the chat window to act on real systems. These collide directly at Snowflake, where you can either use the warehouse's built-in AI or point an external agent like Claude at it through a connector. The pivot point is the Model Context Protocol, an open standard that lets an AI client call external capabilities as named tools. The right way to think about this is that the question is not "which can do more analytics," but "where should the AI actually run, and how much should it reach beyond Snowflake."

Over the next few years, the interesting competition is not over individual features but over *location of intelligence* and *control of the tool surface*. As warehouses embed more AI, the leverage point shifts from "can the model do natural-language querying" (increasingly table stakes) to "who decides which tools an agent may call, and where does orchestration live." That reshapes moats and roles. A platform that keeps AI close to governed data competes on data proximity and built-in controls; an external-agent layer competes on its ability to coordinate across many systems at once. Expect governance to become a tool-permissioning discipline: instead of only managing who can see which rows, teams will manage which capabilities an agent is even allowed to invoke. The durable question is whether the center of gravity sits inside the data platform or in an external orchestration layer that treats every platform, Snowflake included, as one tool among many.

2. The Big Idea and How It Works

Think of your governed data as a secure kitchen. **Cortex is the chef who works inside that kitchen**, cooking right next to the ingredients and never carrying them out. **Claude-plus-MCP is a skilled chef standing outside**, who can call into the kitchen through a controlled service window, request specific dishes, and also coordinate with the bakery and the bar next door. Old world: you exported data to wherever your AI lived. New world: either the AI moves into the data's house (Cortex), or an external agent reaches in through a defined set of tools (MCP). Crucially, the outside chef can order from the inside chef, so these are not opposites.

First, it helps to know that "Snowflake Cortex" is not one product but a family of capabilities. It includes Cortex Analyst, which turns natural-language questions into SQL over structured data; Cortex Search, which powers retrieval-augmented generation (pulling relevant passages from unstructured text so a model can answer with grounded context) over documents; Cortex Agents, which orchestrate multi-step work inside Snowflake; and a set of AI Functions you call directly in queries.

- **The Cortex path.** These run as serverless functions invoked *inside* Snowflake, so the AI executes data-proximate, within the platform's perimeter. From the user's perspective: you ask a question in natural language, Cortex Analyst generates and runs SQL or Cortex Search retrieves passages, and an answer comes back, all without data leaving the warehouse.
- **The MCP path.** An external client, here Claude, connects to a Snowflake MCP server. That server exposes Snowflake capabilities as callable tools, and tellingly, it can expose the *same* Cortex services (Analyst, Search, Agents) plus custom tools and direct SQL execution. From the user's perspective: you ask Claude something, Claude decides which exposed tool to call, the server runs it against governed Snowflake data, and results flow back into Claude's reasoning. Because the toolset is explicit, an implementation can withhold a tool such as direct SQL execution and constrain the agent to a tool backed by a semantic view (a business-friendly layer that maps warehouse tables to concepts like "customers" and "orders"), a real tool-level governance lever for the external client.

3. Implications and Tensions

The headline change is that **most core analytics and retrieval use cases are available in both paths**. Natural-language querying, search, agentic retrieval: reachable whether the AI runs inside Cortex or whether Claude calls Cortex through MCP. So the choice is rarely about feature availability; it is about how and from where those features get invoked.

What MCP adds. Two things Cortex alone does not give you. First, cross-system orchestration: an external agent can query Snowflake and then act in another tool, coordinating across systems that are not Snowflake. Second, fine-grained, tool-level gating for that external client, deciding exactly which capabilities the agent may invoke.

Concrete, near-term. Teams are already deploying a chat-based client connected to a Snowflake MCP server so analysts can ask questions in plain language without writing SQL, with the server enforcing which tools are available to that session. You can expose Cortex Analyst and Cortex Search as tools while deliberately not exposing raw SQL execution.

Medium-term, directional. If external agents become the default front end, expect more workflows where Snowflake is one governed tool inside a larger agentic loop, with the agent reaching into the warehouse for facts and out to other systems to act on them. Teams that already keep data and controls in Snowflake gain the most by starting native and adding the external layer only when orchestration demands it.

In-database vs. external execution. Running AI next to data favors proximity and built-in controls; running an external agent favors flexibility and cross-system reach. Reasonable people weight these differently depending on whether their data and governance already live in Snowflake.

Governance boundary. Snowflake enforces access controls and logging at query time through role-based access control (which assigns permissions to users and roles), masking, and audit logs, and tool gating lets you withhold capabilities like SQL execution. But withholding one tool does not by itself prove an agent is read-only, that depends on the full set of tools exposed. Governance is respected for what is queried, not automatically inherited by whatever the external client does afterward.

What "Claude code" means here. The question names Claude Code, Anthropic's terminal-based agent oriented toward software development. The implementations most commonly documented against Snowflake

MCP servers instead use Claude Desktop, a chat-based client. The tool surface is similar, but the capability profile and typical user differ enough that the answer can shift depending on which product is actually in scope.

Benchmarks and cost. It is intuitive that data-proximate steps benefit from Cortex while cross-system steps benefit from external orchestration, but no public performance, credit, or token comparison exists for this matchup, and the trade-off is workload-dependent. Treat efficiency claims as unsettled.

Either/or framing. The biggest trap is treating these as rivals. Because an external agent can connect via MCP to a tool that is itself a Cortex service, the practical pattern is often *both*: external orchestration wrapped around in-database AI.

4. Hooks, Takeaways, and For the Nerds

A handful of lines that travel well in conversation:

- "The real difference isn't features, it's *where the model runs*: inside Snowflake's perimeter, or as an external agent reaching in."
- "People forget Cortex isn't one product. It's Analyst, Search, Agents, and AI Functions."
- "An MCP client can call the exact same Cortex services as tools, so most analytics use cases exist in both paths."
- "MCP's real add is cross-system orchestration plus tool-by-tool gating, not new analytics."
- "Withholding the SQL tool isn't a read-only guarantee unless you know the whole tool surface."

If you remember three things. The material difference is architectural: AI running in-database (Cortex) versus an external agent reaching in (Claude + MCP), not a gap in features; Most analytics and retrieval use cases exist in both paths because MCP can expose the same Cortex services as tools; this matters because it kills the false either/or; Watch the tool surface: what an external agent can and cannot do depends entirely on which tools the MCP server exposes.

For the nerds. The crux is the tool boundary. An MCP server exposing Snowflake can surface Cortex Analyst, Cortex Search, and Cortex Agents alongside custom tools and an `execute_sql` capability. Whether an external agent is effectively constrained is a property of that *entire* exposed surface, not of any single denied tool. Withholding `execute_sql` while exposing a semantic-view-backed tool is a legitimate design choice, but it is a design choice, not an inherent read-only mode. The current canonical path is the Snowflake-managed/hosted MCP server, and the older community server is explicitly no longer supported; note, though, that the "fully managed, no-infrastructure" framing reflects the documented architecture and likely roadmap rather than a confirmed, branded, generally available product. Note the architecture's honest limit: Snowflake governs access at query time, but once results return to an external client, downstream handling sits outside the perimeter. MCP respects governance for what gets queried; it does not extend that governance end-to-end into the agent's later actions. The open frontier is how much an external client can automatically *discover*, such as schemas, semantic models, and available agents, versus only what the server explicitly chooses to expose.

WHERE SOURCES DISAGREE

What "Claude code" denotes in this question

One reading takes it as Claude Code, Anthropic's terminal-based command-line agent that reads codebases, edits files, and runs tests, which suits agentic software development. Another account, drawn from the implementation walkthroughs, documents Claude Desktop, a chat-based MCP client, as the agent actually connecting to Snowflake — a materially different capability profile

Unresolved: the sources document both products against the same Snowflake MCP tools and do not settle which the question intends

What "just using Snowflake Cortex" scopes to

One reading treats Cortex as a single comparable product with its own use-case boundary. The sources instead consistently describe Cortex as multiple distinct features (Analyst, Search, Agents, AI Functions) with no single standalone "Snowflake Cortex" feature set, so the comparison's framing shifts depending on which Cortex capability is meant

Unresolved: the available evidence describes a family of capabilities rather than one product

Whether the "AI for data engineers vs. AI for product engineers" split is a real boundary

One reading, from opinion commentary, frames Cortex as serving data engineers and external agents as serving product engineers. Other vendor and third-party material show Cortex used by analysts and business users via natural-language and BI interfaces, and external MCP clients front-ending Snowflake for non-developers — blurring the line

Unresolved: the persona split appears as a commentator's heuristic rather than a documented rule, and the hybrid pattern undercuts a clean division

Whether a Snowflake-operated, "no-infrastructure" managed MCP server is a GA, branded product

One reading, from Snowflake documentation, describes a managed MCP server that exposes Cortex tools with built-in OAuth and no separate infrastructure to deploy. Other public product materials describe external-agent integration more generically (APIs, partners, custom services) without a clearly branded, SLA-backed managed MCP offering

Unresolved: current public materials do not consistently confirm the managed-server framing as settled, generally available product versus architecture-plus-roadmap

Cost and performance trade-offs of native Cortex vs. MCP-mediated orchestration

One reading would expect native in-database execution to be more efficient for data-proximate tasks; another would expect external orchestration to be competitive for multi-step or cross-system work

Unresolved: no benchmark, credit, token, or latency data exists in the available evidence

Whether MCP can automatically surface semantic models and "agent discovery" to an external client

One reading describes the external agent gaining live, structured access to catalog, schema, semantic models, and Cortex Agent discovery within its reasoning loop. Another account notes that what an MCP client can see depends on the specific tools the server explicitly exposes, and public docs do not confirm automatic discovery of all semantic models or agents as an inherent property

Unresolved: the evidence supports this as a possible designed capability, not a documented automatic one