MCP SERVER | NO CODE | CLOUD HOSTED

SQL Parser AST Engine MCP

Analyze Query Structure Before Running Code

SQL Parser AST Engine provides deep, programmatic analysis of any SQL query. Instead of just running code, this MCP breaks down the syntax into an Abstract Syntax Tree, revealing every table, column, and join condition. It's essential for security teams checking for unauthorized data access or DevOps engineers validating complex database migrations across 15+ dialects like PostgreSQL, BigQuery, and Snowflake.

A+ Quality Score 100/100

sql-parsing | ast | query-analysis | security-firewall | database-security | code-inspection

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/sql-parser-ast-engine

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

SQL Parser AST Engine MCP

1 tools available
Cloud-hosted on Vinkius

When you deal with SQL queries from user input, the biggest risk isn't just bad syntax; it's what might be hidden inside. This MCP doesn't execute your code. Instead, it takes a raw query string and converts it into an Abstract Syntax Tree (AST). Think of the AST as a detailed blueprint of the query, showing exactly how every piece connects—every table reference, every column name, every join type, and even nested subqueries.

This structured view is critical for governance. Your agent can now inspect the query's intent without ever risking execution. It lets you programmatically check if a query accesses tables outside of an allowed list or if it contains malicious patterns like injection attempts. Because this MCP supports over 15 major SQL dialects, from MySQL to Snowflake, you get one reliable source for analyzing database language across your entire stack. You'll find this engine cataloged and managed easily on Vinkius, giving your AI client a single connection point for all your parsing needs.

Core Capabilities

01 — Detecting unauthorized table access

The MCP analyzes the query to see exactly which tables are referenced, allowing you to enforce data governance rules before execution.

02 — Extracting schema details

It lists every column and table name used in a query, providing fully qualified names necessary for auditing or documentation generation.

03 — Validating SQL syntax fidelity

You can parse an existing query into an AST and then reconstruct valid SQL from that tree structure to ensure full compatibility across different database dialects.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/sql-parser-ast-engine** — connect your AI agent in three steps.

01 Feed the MCP a raw SQL query string, whether it's user input or a script segment needing review.

02 The engine processes the text and generates a structured Abstract Syntax Tree, breaking the entire query into inspectable components like JOIN clauses and column lists.

03 Your agent receives this structured data, which you can then pass to other tools—for example, checking if any extracted tables are on an approved list.

The bottom line is that instead of treating SQL as just text, the MCP gives your agent a machine-readable blueprint of what the query actually intends to do.

Built For

This is for technical roles dealing with database logic and automation. Think security auditors who need proof of data access control, or backend developers debugging complex multi-stage queries that fail in production.

Security Engineer

You use the MCP to automatically vet all incoming API query requests for injection vectors or attempts to breach restricted tables.

DevOps Engineer

You run the parser against migration scripts before deployment, ensuring complex DDL statements are valid and complete across different database versions.

Data Governance Analyst

You process user-generated queries to automatically generate a list of all referenced data assets, improving compliance reporting.

What Changes When You Connect

01 Security checks are airtight. By using the `parse_sql` tool, you don't trust the query text; you inspect its underlying structure for unauthorized operations or injection patterns.

02 Data governance compliance becomes automated. You can use this MCP to extract and list every table and column referenced in any given SQL query, essential for auditing data access rights.

03 It handles database diversity. Since it supports 15+ dialects—including Snowflake, MariaDB, and SQLite—you get one tool that works reliably across your entire mixed-environment stack.

04 Debugging complex logic is easier. You can parse a working query into an AST, then rebuild SQL from the tree to spot subtle syntax differences between dialect versions.

05 The analysis happens safely. The MCP reads the structure; it never executes the code. This means you get deep insight without any risk of running malicious or faulty queries.

Real-World Applications

Auditing a third-party API query

A marketing team submits an ad copy that includes an SQL snippet for data extraction. You feed the snippet to your agent, which uses `parse_sql` to verify that the query only touches the 'leads' and 'campaigns' tables, flagging any attempt to access internal HR records.

Validating a cross-platform migration

Your DevOps pipeline needs to migrate a schema from PostgreSQL to BigQuery. Before deployment, you run the `parse_sql` tool on critical DDL statements across both dialects to ensure all necessary clauses and types are correctly represented in the AST.

Building a query validation layer

You're building an internal data portal where users submit custom queries. Your agent intercepts every request, using `parse_sql` to check for disallowed functions or table joins before sending it to the database.

Analyzing user-submitted search logic

A client gives you a complex SQL query meant for their analytics dashboard. You use the MCP's parsing capabilities to extract all referenced columns, giving your data governance team an immediate list of assets needing review.

Patterns to Avoid

Treating queries as pure text

✗ AVOID

Relying on regex or simple string matching to check for keywords like 'DROP TABLE' or 'SELECT *'. These methods are easily fooled by subqueries, comments, or obfuscated syntax.

✓ INSTEAD

Use the `parse_sql` tool. It builds a formal Abstract Syntax Tree that understands the query's structure, making it impossible to hide unauthorized operations within comments or complex clauses.

Running queries in a sandbox

✗ AVOID

Executing the query against a test environment just to see if it throws an error. This is slow, expensive, and still requires running potentially harmful code.

✓ INSTEAD

Use `parse_sql` first. Analyze the AST locally to identify schema violations or security risks without ever touching a live database connection.

Assuming dialect consistency

✗ AVOID

Writing validation logic based solely on PostgreSQL syntax, only to have it fail when deployed against a Snowflake data warehouse.

✓ INSTEAD

This MCP supports 15+ dialects. Use `parse_sql` with the specific target dialect specified to validate the query structure correctly for your intended database environment.

The Right Fit

Use this MCP if you need structural certainty about a query, not just its execution result. You must use it when building security firewalls, enforcing data governance, or validating cross-dialect code migrations. If the goal is to simply *run* a known good query, then no specialized parser tool is needed. However, if your process involves analyzing user input, vetting external scripts, or checking for potential vulnerabilities—like detecting an unauthorized access attempt in a subquery—you need this deep parsing capability. Don't use this MCP if you just need to count rows; it's about understanding the *intent* and *structure*, not the data payload.

The Pain of Manual Query Audits

Right now, when a query looks suspicious or needs auditing, you're stuck manually reading long blocks of code. You copy the SQL into a text editor, hunting for keywords like 'DROP', 'UNION', or specific table names. If the malicious intent is hidden in a subquery nested three layers deep, or disguised by comments and complex JOINs, your manual review fails instantly.

With this MCP, you feed that suspicious query directly to the parser. It immediately generates an Abstract Syntax Tree—a clean, structured object that exposes every single element. You get an objective list of all tables and columns used, regardless of how complicated or obfuscated the original text was.

How the SQL Parser AST Engine Gives You Structural Proof

The manual effort disappears because you no longer need to hunt for keywords. Instead, your agent uses the `parse_sql` tool to generate a verifiable blueprint. It gives you an explicit list of all referenced data assets, allowing immediate comparison against your allowed schema lists.

It’s a fundamental shift from guessing what the code might do to knowing exactly how it's built. You move from reactive security patches to proactive structural enforcement.

SQL Parser AST Engine: 1 Tool

The listed tool allows you to take any SQL query text and convert it into a detailed, machine-readable Abstract Syntax Tree for deep analysis.

#	TOOL	DESCRIPTION
01	`parse_sql`	Sends an SQL query and receives its Abstract Syntax Tree (AST) structure, along with lists of all tables, columns, and clauses used. Supports major dialects like MySQL, PostgreSQL, and BigQuery.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U A user submitted this SQL query through our API. Parse it and check if it accesses any tables beyond 'orders' and 'products'.

Tables extracted: orders, products, users. ⚠️ 'users' table access detected — not in allowed list.

U Extract all columns referenced in this BigQuery analytics query for our data governance audit.

14 columns extracted across 3 tables with full table.column qualified names.

U Validate this PostgreSQL migration query for syntax errors before deploying to production.

AST parsed successfully — no syntax errors. Query is valid PostgreSQL.

Frequently Asked Questions

01 **Can the SQL Parser AST Engine handle dialects I haven't used before?**

Yes, this MCP supports 15+ major dialects, including MySQL, PostgreSQL, BigQuery, and Snowflake. You specify the dialect when calling `parse_sql` to ensure accurate parsing for your environment.

02 **Does using the SQL Parser AST Engine execute the query against my database?**

No. This MCP analyzes the syntax structure (the Abstract Syntax Tree) and never executes the code. It's purely a reading and validation tool, making it safe for use with untrusted input.

03 **What is an Abstract Syntax Tree in relation to SQL?**

The AST is a structured representation of the query's logic. Instead of seeing text like 'SELECT * FROM t1 JOIN t2', you receive data showing: 'Operation: SELECT; Target Columns: (*); Source 1: (t1); Join Type: INNER; Source 2: (t2)'. This structure is what makes it useful.

04 **Is this better than just using a standard database client for validation?**

Yes. A database client only validates against its own engine rules. The MCP provides an external, programmatic audit layer that can check the query's structure against your custom governance policies (e.g., 'This user cannot access tables X, Y, and Z').

05 **Does parse_sql extract only table names or also column details?**

The `parse_sql` tool extracts both. It provides lists of all referenced tables and, critically, the fully qualified names for every single column used in the query.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"sql-parser-ast-engine": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

SQL Parser AST Engine is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by SQL Parser AST Engine. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	SQL Parser AST Engine MCP
Server ID	`019e38f1-f2c9-707d-9e5d-e2e6c57a179b`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/sql-parser-ast-engine.