ChatGPT

Modern Open-Source Database Query Tools (Late 2025 – Present)

In the past year (since September 2025), several open-source database query tools and data table technologies have gained popularity. These tools emphasize robust configuration, consistency across environments, and high performance (often using Rust or Lua for speed). Below, we highlight some of the most notable ones, including their key features and how they fit into a NixOS/Pixi/Nushell setup. Short descriptions and bullet points are provided for clarity.

RSQL – Universal CLI for Data Queries (Rust-Based)

RSQL is a modern, feature-rich command-line SQL client written in Rust. It provides a **uniform interface** to query numerous data sources (databases, file formats, cloud services) using standard SQL [1]. This means you can use one tool to query everything from PostgreSQL to CSV files with consistent syntax, which greatly aids configuration consistency and scripting in a Nushell environment.

- **Multi-Source Support:** Works with 20+ databases and formats (Postgres, MySQL, SQLite, DuckDB, CSV, Parquet, JSON, etc.) via a single CLI interface [1] [2]. For example, `rsql` can query a local CSV or a remote Snowflake DB with equal ease.
- **Consistent SQL Interface:** Always uses ANSI-standard SQL for queries, abstracting away source-specific quirks. This uniformity ensures your query "language" remains the same symbol-by-symbol, whether querying a database table or a text file.
- **Rich Configuration & Output Options:** Supports setting default output format (JSON, CSV, table, etc.) and other options via flags for consistent results [3]. It offers interactive features (syntax highlighting, autocompletion, history) and multiple output formats (ASCII/Unicode tables, JSON, XML, YAML, Markdown) for flexible tooling [4] [5].
- **All-in-One Convenience:** Includes an embedded DuckDB and even an **embedded Postgres mode**, so you can run SQL queries on the fly without external DB setup [6] [5]. This makes it an "all-in-one" Swiss-army knife for data querying – ideal for scripting in an **agentic coding** workflow where a CLI tool can be invoked by an AI agent to retrieve or join data.

SurrealDB – Multi-Model Database for AI Agents (Rust-Based)

SurrealDB is an open-source multi-model database engine built in Rust, designed to **unify multiple data models into one system** [7]. It supports document, graph, relational, time-series, geospatial, and key-value data – all accessible with a single SQL-like query language called **SurrealQL** [8]. SurrealDB emphasizes **consistency** and rich features, making it popular for modern applications and AI agent contexts.

- **All-in-One Multi-Model:** Combines capabilities of document stores, graph databases, full-text search, and vector (embedding) search in one engine [8]. This means you can perform complex

queries (e.g., relational joins, graph traversals, vector similarity) within one database, simplifying architecture.

- **AI-Native & "Symbol-Level" Detail:** *"The multi-model database for AI agents"* is how SurrealDB is branded [9]. It's built to handle the **fine-grained data needs of AI workflows**, allowing agent applications to query knowledge graphs, embeddings, and relational data in one place. SurrealDB's support for vector and semantic queries gives AI agents a kind of "symbolic" or context-rich querying ability (e.g., finding records by similarity in vector space).
- **Powerful Developer Features:** SurrealQL syntax is intuitive, blending SQL with graph pattern matching – you can start schemaless and then add schema for consistency [10] [11]. It also supports real-time subscriptions and server-side functions/triggers, enabling detailed **diagnostics and reactive behavior** (agents can subscribe to data changes or run procedures on events).
- **Deployment Flexibility:** SurrealDB is a single lightweight Rust binary – it can run embedded in an app, at edge (WASM in browser), as a standalone server, or in a cluster [12]. This makes configuration straightforward across environments (e.g., easily integrated into NixOS setups). Its native ACID compliance ensures strong consistency across all these modes [13].

Sources: SurrealDB unifies document, graph, time-series, relational, etc., in one engine [8]. It's used for AI agents and knowledge graphs, and runs as a single Rust binary (embedded or distributed) [12].

DuckDB – Embedded Analytics Database (C++ with SQL)

DuckDB has rapidly become one of the most popular open-source SQL query engines for analytics. It's a high-performance **columnar relational database** designed for **OLAP** (analytical) queries on large datasets [14]. DuckDB runs **in-process** (like SQLite for analytics), making it easy to deploy (no server needed) and ideal for both CLI use and integration into tools.

- **Lightweight & Fast:** DuckDB is an embeddable library/CLI that runs inside your application or shell without network overhead [15]. Despite its small footprint, it uses a vectorized query execution engine for speed on large data [15].
- **Feature-Rich SQL:** It supports a broad SQL dialect (window functions, complex joins, etc.) and can query data in situ from files. This means you can point DuckDB at a Parquet or CSV file on disk and run full SQL queries immediately.
- **File and Table Format Support:** DuckDB works with many data formats out-of-the-box – CSV, JSON, Parquet, even Apache Iceberg table format, etc. [16]. For example, you can do `SELECT * FROM 'data.parquet'` or even join a Parquet file with a CSV. This makes it extremely convenient in a **Nushell** environment for quick data analysis.
- **Up-to-Date and Open Source:** DuckDB's development is active; it reached version 1.0 in mid-2024 and continues to release updates. The latest Long-Term Support version was **v1.4.3 (released Dec 9, 2025)** [17], indicating a mature and stable platform. It's MIT-licensed and widely adopted, ensuring community support.
- **Config Consistency:** Because it's a self-contained binary, configuration is minimal – no services or clusters to manage. For consistent results, you can set PRAGMA settings or use DuckDB's extension system. Its behavior is deterministic, which is great for reproducible analysis in different environments (including NixOS).

Apache Iceberg – Open Table Format for Data Lakes

Apache Iceberg isn't a query *tool* per se, but an important open-source **table format** that has become a cornerstone for querying large datasets consistently. Iceberg is a high-performance table storage layer for data lakes, providing **ACID transactions, schema evolution, and partitioning** on files (Parquet/ORC/Avro) in object storage [18]. Since late 2025, Iceberg has been in the spotlight as a way to ensure data **consistency** across various query engines.

- **Unified Table Layer:** Iceberg lets you treat a collection of data files as a single **relational table** with full SQL semantics. Multiple engines (Spark, Trino, Flink, DuckDB, etc.) can read these tables consistently. This solves the "siloed data" issue by eliminating format incompatibilities [19].
- **Key Features for Consistency:** It supports **schema evolution** (you can add/drop columns without breaking readers), **time travel** queries (query data as of an older snapshot), and **hidden partitioning** (automatic partition management) [18]. These ensure that as your data changes, all tools see a consistent view without manual reconfiguration.
- **Performance and Scale:** Designed for petabyte-scale analytics, Iceberg tables avoid the pitfalls of older Hive tables. It keeps metadata on snapshots and uses techniques like metadata pruning for faster queries. This means even as data grows, query tools like Spark or DuckDB can plan queries efficiently on Iceberg tables.
- **Open and Popular:** Iceberg is fully open-source (Apache License 2.0) and became an Apache top-level project. By 2025 it's considered a **de facto industry standard** for data lakehouse storage [18], with broad support from cloud vendors and data platforms. For an environment like NixOS, Iceberg's Java libraries or CLI can be used, but more interestingly, engines like DuckDB or Trino that run on NixOS can directly query Iceberg tables, benefiting from its consistency guarantees.

Sources: Iceberg is an open-source high-performance table format for analytic data, offering schema evolution, time travel, and other features for unified data architecture [20] [18]. It has become a critical standard supported by many big data systems [18].

Tarantool – Lua-Powered Database and Application Server

Tarantool is a unique open-source database that combines an **in-memory NoSQL database** with a built-in **Lua application server**. It's included here for its use of Lua (a fast embedded language) and its focus on performance and consistency. Tarantool keeps all data in RAM (with disk persistence via write-ahead logs) for extreme speed, and allows you to write server-side logic in Lua, blurring the line between application and database [21] [22].

- **High Performance & Low Latency:** Tarantool was designed for high-throughput, low-latency scenarios (e.g. telecom messaging, cache, queues). Data is memory-resident by default, yielding predictable fast access [23]. It uses cooperative multitasking (coroutines) rather than locking, which aligns with Lua's strengths for concurrency [24] [25].
- **Lua for Logic:** Rather than use external stored procedures or ORMs, Tarantool lets developers write their business logic *inside* the database in Lua. Essentially, Tarantool **"is not just a database server – it's a programming language"**, acting as a Lua runtime with persistence [26]. This gives symbol-level control over data handling – you manipulate data structures and queries directly via Lua APIs, enabling very detailed, custom operations (great for diagnostics or complex transactions).

- **SQL Support:** Initially a NoSQL (key-value) and Lua-only database, Tarantool **introduced an SQL query interface in 2019** that complies with standard SQL features [27]. Now you can run SQL queries on Tarantool tables (which are called spaces) if needed, making it more accessible as a general database. The SQL layer integrates with its NoSQL engine, so you get the flexibility of both.
- **Consistent Config & Deployment:** Tarantool runs on Linux (it's popular on Debian/Ubuntu and can be managed in NixOS as well). Configuration is done via a Lua config file, which means you script your setup (ports, replication, users, etc.) in Lua – quite consistent and versionable. Because the same Lua code can define config and runtime logic, there's less context switching, and an "agentic" coding agent could potentially reason about the config as code. Tarantool's small footprint (one binary, no JVM) fits well in containerized or Nix environments.

Sources: Tarantool is an in-memory DB + Lua application server in one, offering an interactive Lua environment with ACID storage [21]. As of 2019 it also provides a SQL interface adhering to standard SQL features [28], making it a hybrid solution for fast data processing.

MCP Toolbox for Databases – Connecting AI Agents to SQL (Trend 2025)

Finally, a new tool relevant to "agentic vibe coding" is **MCP Toolbox for Databases**. This isn't a database or traditional query client, but an open-source server/proxy that uses Anthropic's **Model-Context Protocol (MCP)** to let AI agents safely query databases. In late 2025 it gained attention as a way to integrate LLM-based agents with structured data systems [29].

- **Standardized Protocol for AI→DB:** MCP Toolbox acts as a middle layer where an AI (like an LLM-based agent) sends high-level requests, and the toolbox translates them to actual SQL queries to run on your database (PostgreSQL, MySQL, etc.) [30] [31]. It essentially gives AI a controlled, symbolic access to the database – the AI doesn't execute raw SQL directly but goes through this standardized interface.
- **Safe and Easy Integration:** By using a defined protocol (MCP), it eliminates the need for custom integration code for each AI application [30]. This is important for consistency and security – you can enforce what an agent is allowed to do at the MCP server level. For an "agentic coding" scenario, you could spin up the MCP Toolbox on NixOS and let the agent ask for data (the toolbox ensures the queries are well-formed and safe).
- **Developed by Google, Open Source:** Originally called the "GenAI Toolbox for Databases," it was open-sourced by Google under Apache 2.0 and quickly rose in GitHub's trending list [29]. Its popularity signals a trend of bridging AI and databases in a standardized way. For a developer, this means the configuration is largely about pointing the toolbox to your DB credentials and letting the AI connect – consistent across different backends.
- **All-in-One for Agents:** While not a traditional query tool for human developers, MCP Toolbox is an **all-in-one solution for agent-based querying**. It handles multiple DB types and can be extended to new ones, giving a unified front-end (the MCP API) to various data sources. This aligns with the "one interface to rule them all" philosophy, but oriented to AI usage. In effect, it's the counterpart to human tools like RSQL, but for AI.

Sources: "MCP Toolbox for Databases is an open-source MCP server that allows developers to easily and safely connect AI systems and their large language models to structured data... eliminating the need to develop custom integrations." [30] It supports linking AI agents to multiple SQL databases via a standardized protocol [32],

and was released by Google (formerly GenAI Toolbox), now Apache 2.0 licensed and trending on GitHub [29].

Each of these tools and technologies addresses a facet of modern data querying: **RSQL** for unified CLI access, **SurrealDB** for multi-model and AI-ready workloads, **DuckDB** for local analytics, **Iceberg** for cross-engine table consistency, **Tarantool** for Lua-speed and embedded logic, and **MCP Toolbox** for AI agent integration. All are open source and emphasize configuration consistency and performance. In an environment built on NixOS with Pixi and Nushell, these tools can be installed and configured reproducibly, providing a powerful arsenal for both developers and AI coding agents to query and manage data at a highly detailed (even "symbolic") level.

Sources:

- RSQL – multi-source SQL CLI (Rust) [1] [5]
- SurrealDB – Rust multi-model DB for AI (SurrealQL, vectors, graph) [8] [12]
- DuckDB – embedded analytic DB (OLAP, file support, v1.4.3 in Dec 2025) [16] [17]
- Apache Iceberg – open table format (schema evolution, time travel for lakehouse) [20] [18]
- Tarantool – in-memory Lua DB/app server, optional SQL interface [21] [28]
- MCP Toolbox for DB – AI agent database interface (MCP protocol) [30] [29]

---

[1] [2] [3] [4] [5] [6] GitHub - theseus-rs/rsql: Command line SQL interface for relational databases and common data file formats
https://github.com/theseus-rs/rsql

[7] [8] [10] [11] [12] [13] Introduction
https://surrealdb.com/docs/surrealdb/introduction

[9] The ultimate multi-model database | SurrealDB
https://surrealdb.com/

[14] [15] [16] [17] [18] [19] [20] [29] [30] [31] [32] The 10 Coolest Open-Source Software Tools Of 2025
https://www.crn.com/news/software/2025/the-10-coolest-open-source-software-tools-of-2025

[21] [24] [25] [27] [28] Tarantool - Wikipedia
https://en.wikipedia.org/wiki/Tarantool

[22] [23] [26] Tarantool: A Lua-based database engine | Opensource.com
https://opensource.com/business/16/4/tarantool-lua-based-database-engine