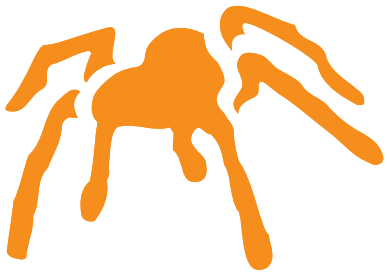

WHITE PAPER

The Relational Database in an AI World:

How Mimer SQL Powers Trusted, Standards-Compliant AI Data Access

Mimer Information Technology AB
Uppsala, Sweden • mimer.com
April 2026

Executive Summary

Artificial intelligence is reshaping how organisations create value from their data. Large language models, autonomous agents, and agentic workflows are moving rapidly from experimental projects into production systems — and they all need one thing above all else: reliable, governed, structured access to data.

The relational database, far from being displaced by the AI wave, has become more central than ever. Its foundational principles — structured schemas, ACID transactions, declarative queries, and rigorous access control — are precisely the properties that make AI outputs trustworthy and auditable. Against this backdrop, Mimer SQL occupies a uniquely compelling position: it is the only database management system that delivers 100% conformance to the international ISO/ANSI SQL standard while simultaneously being deployable from enterprise servers to deeply embedded IoT devices running on real-time operating systems.

This white paper explores the role of the relational database in AI-centric architectures, examines the Model Context Protocol (MCP) as a new integration layer for AI agents, and explains why Mimer SQL — with its unmatched standards compliance, minimal footprint, zero-maintenance architecture, robust security model, and MCP connectivity — is ideally positioned to serve as the trusted data hub at the heart of AI deployments of any scale.

Key Thesis

In an AI world, the relational database does not become less important — it becomes the governance layer that makes AI trustworthy. Mimer SQL's unique combination of 100% SQL standard compliance, exceptionally small and scalable footprint, and MCP connectivity makes it the ideal choice for organisations embedding AI into any environment, from cloud servers to edge devices.

1. The AI Data Challenge

1.1 From Data Abundance to Data Trust

The past decade saw organisations invest heavily in collecting data. Data lakes, warehouses, streaming pipelines, and NoSQL stores proliferated as teams raced to accumulate the raw material that machine learning would transform into competitive advantage. The results were mixed. Data abundance did not automatically produce trustworthy AI outputs. Models trained or grounded on poorly governed, inconsistently structured, or stale data produced recommendations that were difficult to explain, audit, or act upon with confidence.

The emergence of large language models as general-purpose reasoning engines has sharpened this problem. An LLM that can hold a natural-language conversation about a company's operations is only as useful as the data it can access. If that data is unstructured, ungoverned, or inaccessible in real time, the model's output is correspondingly unreliable. The industry is learning that AI capability and data governance are not separate concerns — they are inseparable.

1.2 What AI Agents Need from a Database

A new generation of AI components has emerged: the AI agent. Unlike a batch model that runs periodically, an agent operates autonomously, calls tools, retrieves data, and iterates toward a goal. Agents interact with databases in fundamentally different ways than traditional applications:

- Exploratory access: agents discover what data is available before formulating queries, requiring rich schema introspection capabilities.
- Natural language to query translation: agents translate human intent into structured queries, introducing the risk of syntactically incorrect or semantically wrong SQL if not properly controlled.
- Read safety by default: agents must be constrained from performing destructive operations unless explicitly and deliberately authorised.
- Auditability: every data access made by an agent must be traceable for compliance and debugging purposes.
- Security isolation: different agents, users, and applications must operate within precisely scoped permissions, enforced at the database level.
- Consistency guarantees: agents that read and act on data must see a consistent snapshot; stale or partially-committed data can lead to incorrect and potentially harmful decisions.

These requirements are not new — they are the foundational promises of the relational database. ACID transactions, role-based access control, declarative SQL, and schema-enforced data integrity are the exact mechanisms that make relational databases the right foundation for AI data access.

1.3 The Risk of Bypassing Structure

Some AI architectures attempt to bypass structured databases entirely, allowing LLMs to generate and execute arbitrary SQL directly, or to query flat files and object stores without governance. This introduces serious risks:

- SQL injection and unintended data modification when generated queries bypass parameterisation or are executed with overly permissive credentials.
- Non-deterministic query behaviour — the same natural language question may produce different SQL across model calls, leading to inconsistent results.
- No enforcement of business rules, referential integrity, or access policies at the data layer.
- Inability to audit or explain what data was accessed, by which agent, and for what purpose.

The industry consensus is converging on a controlled-access model: AI agents interact with databases through a defined, secured interface layer — not by generating arbitrary SQL against production systems. The relational database, with its mature security and integrity mechanisms, is the natural backend for this model.

2. The Enduring Role of the Relational Database

2.1 Structure as a Foundation for AI Trust

The relational model, first articulated by Edgar Codd in 1970, organises data into tables with defined schemas, enforces relationships through foreign keys, and guarantees transactional consistency through ACID properties (Atomicity, Consistency, Isolation, Durability). These are not legacy constraints — they are the mechanisms that make data trustworthy enough to reason over and act upon.

For AI systems, schema-enforced structure provides several critical guarantees: data type correctness ensures that a date field cannot accidentally contain a string; referential integrity ensures that an AI agent querying order data can trust that every customer reference points to a real customer; consistent transaction snapshots prevent the AI from reasoning over a partially-updated state; and declarative SQL semantics allow AI-generated queries to be validated and optimised before execution.

2.2 SQL as a Universal Interface

SQL has been the standard interface to relational data for over four decades. Its longevity reflects the fact that declarative, set-based query semantics are well-suited to the kinds of data retrieval and aggregation that business applications — and AI agents — require. Every developer, data scientist, and increasingly every AI model trained on code has exposure to SQL.

This universality means that an AI agent capable of generating SQL can, in principle, interact with any SQL-compliant database. The practical reality is more nuanced: vendor-specific SQL dialects, proprietary extensions, and non-standard behaviours mean that SQL written for one database often fails on another. True SQL standard compliance — not merely claimed compliance — is the property that makes this portability real and reliable.

2.3 Security and Governance as First-Class Properties

The relational database was designed from the beginning with multi-user access control in mind. Users and roles, object-level privileges, view-based row and column security, and audit logging are mature, well-understood mechanisms refined over decades of enterprise deployment.

These mechanisms are exactly what AI deployments require. An AI agent should only see the data it is authorised to see. Its ability to modify data should be explicitly granted, not assumed. The relational database's access control model, when properly configured, provides a natural and enforceable security perimeter around data accessed by AI agents — a perimeter enforced at the database level, not dependent on the correctness of application code or the reliability of the AI model itself.

3. The Model Context Protocol: AI Agents Meet Databases

3.1 What Is the Model Context Protocol?

The Model Context Protocol (MCP) is an open standard, introduced by Anthropic in late 2024, that defines how AI applications discover and invoke external tools and data sources. MCP provides a standardised interface — often described as a "universal connector" for AI — that eliminates the need for custom integrations between each AI model and each external system.

An MCP server exposes a curated set of tools that an AI client can discover and invoke. The client does not need to know the implementation details — it calls the tool by name with appropriate parameters, and the MCP server handles the actual operation, including any necessary translation, validation, and security enforcement.

> **MCP Architecture in Three Layers**
> AI Client (Claude, GitHub Copilot, Cursor, or custom agent) → MCP Layer (tool discovery, invocation, parameter validation, read-only enforcement) → Database Layer (secure, validated, parameterised operations). This separation ensures that security policies are enforced at the server level, database credentials are centrally managed, and AI agents cannot perform operations beyond their defined scope.

3.2 Why MCP Is the Right Interface for AI-Database Integration

Previous approaches to connecting AI to databases — allowing LLMs to generate and execute arbitrary SQL — suffered from predictable failure modes: syntax errors, incorrect query logic, security vulnerabilities, and non-deterministic behavior. MCP addresses these problems by exposing a controlled set of operations with built-in guardrails:

- Tool schema validation: MCP tools have defined parameter schemas, preventing malformed requests from reaching the database engine.
- Read-only enforcement at the protocol layer: MCP servers can restrict agents to SELECT operations independently of database-level permissions, providing defense in depth.
- Schema discovery before query execution: agents can explore available schemas, tables, and stored procedures before formulating queries, reducing incorrect assumptions.
- Parameterised SQL internally: MCP servers use parameterised queries, preventing SQL injection regardless of what the AI generates in natural language.
- Centralised audit logging: all tool invocations pass through the MCP server, creating a natural and complete logging point for AI data access.

3.3 The Mimer MCP Server

Mimer Information Technology has developed the Mimer MCP Server, a dedicated MCP implementation that connects AI applications to Mimer SQL databases. The server acts as a secure intermediary, translating natural language-driven tool calls into validated, parameterised database operations — while enforcing Mimer SQL's rigorous user and group security model at every step.

The Mimer MCP Server exposes tools organised into three categories:

Schema Discovery Tools

- list_schemas — retrieves all available schemas in the connected database, enabling agents to understand what data domains are present.
- list_table_names — lists all tables within a specified schema, allowing agents to identify relevant data sources.
- get_table_info — provides detailed schema information and sample rows for specified tables, giving agents the context they need to formulate accurate queries.

Query Execution Tools

- execute_query — executes SQL queries with full parameter support; only SELECT statements are permitted, ensuring that no data modification can occur through the MCP interface.

Stored Procedure Tools

- list_stored_procedures — lists available read-only stored procedures, exposing curated, pre-validated data access logic to agents.
- get_stored_procedure_definition — retrieves the definition of a stored procedure for agent inspection.
- get_stored_procedure_parameters — retrieves the parameter signature, enabling correct invocation.
- execute_stored_procedure — executes a stored procedure with JSON-encoded parameters, supporting complex, encapsulated data access patterns. Only read-only procedures are allowed.

The Mimer MCP Server integrates with leading AI development environments including VS Code with GitHub Copilot, Claude Desktop, and Cursor, and can be deployed as a Python package or Docker container. Connection pool management ensures efficient resource utilisation under concurrent agent workloads, and the server architecture provides a clear separation between the AI layer, the MCP protocol layer, and the Mimer SQL database layer.

> **Mimer SQL as the AI Data Hub**
>
> With the Mimer MCP Server, Mimer SQL becomes the trusted hub of AI data interactions — a governed, standards-compliant repository that AI agents can explore and query safely. The database's mature user and group security model ensures that each agent, application, or user accesses only the data they are authorised to see, regardless of how the natural language query was phrased or which AI model generated it.

4. Mimer SQL: Uniquely Positioned for an AI World

4.1 100% ISO/ANSI SQL Standard Compliance

Mimer SQL has long been recognised as an unofficial reference implementation of the ISO/ANSI SQL standard. This is not a marketing claim — it is a commitment embedded in the product's development philosophy since its origins as a research project at Uppsala University in the 1970s, and it is one that Mimer has maintained consistently across decades of commercial development.

All products in the Mimer SQL family are characterised by 100% conformance to the international SQL standards. In practice, this means:

- AI-generated SQL that conforms to the standard will execute correctly on Mimer SQL without vendor-specific adjustments or dialect translation.
- The full SQL feature set is available — stored procedures (ISO PSM), triggers, views, domains, strong typing, and full Unicode support — even in embedded configurations.
- Applications built on Mimer SQL can be migrated to and from other standard-compliant databases without rewriting SQL logic, preserving investment and enabling architectural flexibility.
- Developers and AI models trained on standard SQL can be immediately productive without learning proprietary extensions or workarounds.

This matters acutely in AI contexts because AI models generate SQL based on their training data, which reflects standard SQL more than any specific vendor's dialect. A database that deviates from the standard introduces a translation layer that adds complexity and creates opportunities for error — exactly the kind of brittleness that production AI deployments cannot afford.

For readers who wish to examine the compliance picture independently, the Wikipedia article on SQL compliance (en.wikipedia.org/wiki/SQL_compliance) provides a feature-by-feature breakdown across the major database systems. Scanning across the Mimer SQL column, it is the only database in the comparison that returns no gaps across the core SQL feature set — a pattern that is immediately visible and that no amount of vendor documentation can replicate.

4.2 Minimal Footprint Without Compromise

Perhaps Mimer SQL's most remarkable technical characteristic is the breadth of its deployment range. From the same kernel, Mimer SQL can be configured to target environments ranging from large enterprise servers to deeply embedded systems with severe resource constraints. A Mimer SQL server executable typically measures in the range of a few to around fifteen megabytes — a fraction of competing enterprise databases — and tailor-made embedded configurations can be reduced further still, making it viable in constrained environments where other full-featured databases simply cannot operate.

Deployment Profile	Significance for AI
Enterprise Server	Full-featured deployment; suitable for cloud and on-premises AI agent backends with many concurrent connections.
Industrial Embedded	Configurable embedded deployment for industrial devices, appliances, SCADA systems, and automotive controllers.
Constrained Embedded	Tailor-made configurations with a small footprint suited for resource-constrained AI edge deployments, without sacrificing SQL correctness or concurrency control.
Real-Time Systems	Native integration with real-time operating systems including QNX on ARM and x86; supports mission-critical latency requirements.
Mobile / Edge Gateway	Deployable on Android and other mobile platforms; previously embedded in millions of mobile devices globally.

This deployment range is highly distinctive in the industry. Very few SQL-standard-compliant database management systems can operate meaningfully in the constrained environments that Mimer SQL targets. This makes Mimer SQL the natural — and in many cases the only viable — choice for organisations deploying AI capabilities at the edge: in IoT gateways, industrial controllers, autonomous vehicles, medical devices, and other environments where a conventional enterprise database is impractical, but where data integrity and governance remain non-negotiable.

4.3 Zero-Maintenance Architecture

Mimer SQL's zero-maintenance design is a deliberate architectural choice with significant practical implications for AI deployments, where database access patterns are driven by natural language input rather than predetermined application logic:

- Automatic database reorganisation: database files are always structured for optimal performance without manual index rebuilds or storage reclamation utilities.
- Non-locking concurrency control: Mimer SQL uses optimistic concurrency control (OCC), eliminating deadlocks and the performance overhead of pessimistic locking. Multiple AI agents and applications can simultaneously read and update the database without contention or queuing.
- Self-tuning kernel: very few tuning parameters are required — primarily cache size and thread count — dramatically reducing operational complexity in production AI deployments.
- No scheduled maintenance windows: unlike databases that require periodic VACUUM, ANALYZE, or reorganisation operations, Mimer SQL maintains itself continuously and automatically.

In AI deployments, these properties are particularly valuable because AI agents generate queries dynamically and unpredictably. A self-tuning, lock-free database is better equipped to handle the variable, concurrent, and sometimes burst-heavy access patterns that AI workloads produce, without requiring a DBA to anticipate and tune for every access pattern in advance.

4.4 Security Architecture

Mimer SQL implements a comprehensive, multi-layered security model that maps naturally onto the access control requirements of AI agent deployments:

- Users and groups: Mimer SQL supports distinct database users and groups with independently scoped privileges, enabling the principle of least privilege for AI agent deployments. Each MCP Server instance operates under a single configured database identity; to isolate agents or agent classes, deploy a separate instance per group, each with its own scoped database user.
- Object-level privileges: tables, views, schemas, and stored procedures can be granted or revoked independently for each user or group, enabling precise access scoping.
- View-based security: complex row- and column-level security policies can be implemented through views, presenting AI agents with precisely the data they are authorised to see, regardless of the underlying table structure.
- Stored procedure encapsulation: complex data access logic can be encapsulated in stored procedures, exposing a defined, auditable interface to AI agents rather than raw table access — reducing the surface area for unintended data exposure.
- Credential isolation via connection pooling: the Mimer MCP Server manages a connection pool with centralised credential management, ensuring that AI clients never handle raw database credentials.

This security model enforces the principle of least privilege at the database level — the appropriate enforcement point, where it cannot be bypassed by application-layer logic errors, misconfigured agent permissions, or prompt injection attacks targeting the AI layer.

4.5 Platform Breadth and API Support

Mimer SQL runs on all major platforms — Windows, Linux, macOS, QNX, OpenVMS, and Android — from the same database kernel. This means that organisations deploying AI at the edge on Linux-based IoT gateways use the same database engine, with the same SQL behaviour and security model, as their enterprise server deployments. Consistent behaviour across the deployment spectrum simplifies development, testing, and governance.

API support covers the full spectrum of development environments relevant to AI and data workloads:

- ODBC, JDBC, and ADO.NET for standard application integration
- Mimer SQL C API and Embedded SQL for C/C++ systems programming
- MimerPy for Python with SQLAlchemy support, directly relevant for AI and data science
- Qt integration for embedded GUI applications
- Node.js for web and desktop applications
- Mimer MCP Server for AI agent connectivity

5. Reference Use Cases

5.1 Enterprise AI Agent with Governed Data Access

In an enterprise setting, AI agents are being deployed to assist with customer service, operations analysis, and decision support. A typical architecture connects an AI client to one or more enterprise data systems through the MCP layer. With Mimer SQL and the Mimer MCP Server, this architecture delivers schema-aware exploration — agents can discover available data without prior knowledge of the database schema — combined with natural language querying through the MCP tool interface, role-scoped access enforced at the database level for each agent class, and a complete audit trail of all agent data accesses.

The Mimer SQL security model ensures that each agent type is provisioned with precisely the access it needs: a customer service agent may have read access to order and customer data, while a financial analysis agent has access to a separate schema. These boundaries are enforced by Mimer SQL's user and group privilege system, not by application-layer logic that could be circumvented.

5.2 Edge AI with On-Device Embedded Database

Edge AI is one of the fastest-growing deployment patterns — AI inference running on IoT gateways, industrial controllers, autonomous vehicles, and medical devices. These environments have strict resource constraints but also strict requirements for data integrity and local data persistence.

Mimer SQL Embedded, with its compact footprint and full SQL support, enables a pattern that very few other SQL-compliant databases can match: an AI inference system running on a constrained device, with a locally embedded database providing persistent storage for sensor readings, operational state, and inference results; concurrent access for multiple processes on the device under full concurrency control; ACID transaction guarantees ensuring that partial writes due to power loss do not corrupt the local database; and synchronisation readiness so that when connectivity is restored, the locally stored data's integrity can be relied upon for upload to central systems.

5.3 Automotive and Mission-Critical AI

Automotive systems represent a demanding deployment context: real-time constraints, zero tolerance for database-induced latency spikes, and long operational lifetimes without maintenance. Mimer SQL being chosen as a persistency layer in Bosch/ETAS product for AUTOSAR Adaptive Platform demonstrates the robustness of its zero-maintenance, non-locking architecture in real-world mission-critical use.

As AI systems are integrated into automotive applications — for diagnostics, predictive maintenance, driver assistance, and fleet data collection — the requirements for the underlying database become even more stringent. Mimer SQL's non-locking concurrency control, native real-time OS support, and self-maintaining design make it the appropriate database engine for AI-integrated automotive and industrial systems.

5.4 AI-Assisted Data Exploration

An immediately compelling use case for the Mimer MCP Server is AI-assisted data exploration — giving business analysts, data scientists, and operations staff the ability to query a Mimer SQL database in natural language through an AI client, without requiring SQL expertise.

The schema discovery tools (list_schemas, list_table_names, get_table_info) allow an AI agent to autonomously understand the database structure before formulating queries. This enables natural language interactions such as querying for unshipped high-value orders, identifying products with declining sales trends, or discovering and invoking stored procedures for compliance reporting — all without the user needing to know the underlying table structure or query syntax.

Throughout these interactions, the Mimer SQL security model ensures that the AI agent accesses only data within its provisioned scope, and the MCP Server's read-only query interface prevents any inadvertent data modification regardless of how the natural language query is phrased.

6. Architectural Principles for AI-Database Integration

Based on the analysis in this paper, the following principles should guide the integration of relational databases into AI architectures:

Principle 1: Enforce Least Privilege at the Database Level

AI agents should be provisioned as database users with the minimum privileges necessary for their function. Access control enforced at the database level cannot be bypassed by application-layer vulnerabilities, misconfigured agent systems, or prompt injection attacks. Mimer SQL's user and group model provides the granularity to implement this principle precisely.

Principle 2: Use a Controlled Access Layer

AI agents should not execute arbitrary SQL directly against production databases. An MCP server or equivalent controlled-access layer should validate, parameterise, and restrict database operations before they reach the database engine. The Mimer MCP Server provides this layer with built-in read-only enforcement and parameterised query handling.

Principle 3: Require Genuine SQL Standard Compliance

The database underpinning AI data access should genuinely conform to the ISO/ANSI SQL standard. This ensures that AI-generated SQL executes correctly and predictably, and that the database can be integrated with any standards-compliant tool or migrated without SQL rewrites. Mimer SQL's 100% standard compliance is the most stringent available in the industry.

Principle 4: Design for Schema Discoverability

AI agents perform better when they can explore the database schema autonomously before formulating queries. Databases should expose rich schema introspection, and MCP servers should surface these capabilities through well-designed schema discovery tools.

Principle 5: Audit Everything

Every data access by an AI agent should be logged with sufficient context to reconstruct what was accessed, by which agent, when, and for what purpose. The MCP server layer is the natural point for this logging — all tool invocations pass through it, making it a complete and reliable record of agent data activity. This architectural separation is itself an advantage: audit responsibility sits in the integration layer, where it can be configured, extended, and connected to enterprise logging infrastructure independently of the database engine.

Principle 6: Match the Database to the Deployment Context

Not all AI deployments run in data centers. Edge AI, embedded AI, and mobile AI require databases that can operate in resource-constrained environments without sacrificing correctness or security. Mimer SQL is uniquely positioned to serve the full range from embedded edge to enterprise server with a consistent kernel, data model, and API surface.

7. Conclusion

The relational database is not a relic of a pre-AI era — it is the governance layer that makes AI trustworthy. As AI agents move into production, the organisations that succeed will be those that ground their AI systems in well-governed, structured, auditable data. The relational database — mature, proven, and now directly integrated with AI agent tooling through standards like MCP — is the right foundation for this.

Mimer SQL occupies a singular position in this landscape. Its 100% conformance to the ISO/ANSI SQL standard — a commitment maintained since the product's origins at Uppsala University and sustained across every deployment profile — ensures that AI-generated queries execute correctly and predictably. Its ability to deploy from compact embedded configurations to full enterprise servers, all from a single kernel, means that organisations can apply a consistent data governance model across the full spectrum of their AI deployments: from cloud-hosted enterprise agents to AI inference on constrained edge devices.

The Mimer MCP Server transforms Mimer SQL into a first-class participant in the emerging AI agent ecosystem. Through the Model Context Protocol, AI clients can safely explore and query Mimer SQL databases using natural language, with the database's mature user and group security model enforcing access boundaries at every layer. The architecture is clean: the AI layer handles reasoning and natural language understanding; the MCP layer provides controlled, validated tool invocation; and Mimer SQL provides the trusted, standards-compliant, governed data store underneath.

For organisations building AI systems that must be trustworthy, auditable, and deployable anywhere — from enterprise data centres to the edge of the network, from high-performance servers to embedded environments — Mimer SQL provides a foundation that no other database can match.

About Mimer Information Technology AB

Mimer Information Technology AB is a Swedish database technology company headquartered in Uppsala, Sweden. Its roots lie in Uppsala University's computer science research of the 1970s. Mimer SQL is deployed in mission-critical systems worldwide, including the UK National Health Service's blood transfusion service and Volvo Cars' production lines in Gothenburg. Mimer is widely recognised as an unofficial reference implementation of the ISO/ANSI SQL standard. For more information, visit mimer.com or developer.mimer.com.