ORACLE

Oracle Deep Data Security

Database-enforced access control for agentic AI, analytics, and enterprise applications

April, 2026 1.0

Purpose of this document

This technical brief explains how to safeguard data access in environments where AI agents, analytics, and applications interact directly with enterprise data. As agentic AI drives dynamic SQL generation and expands the attack surface, traditional controls may fail to keep pace. The document explains why these risks matter and shows how Oracle Deep Data Security enforces fine-grained authorization directly within Oracle AI Database 26ai, where the data resides, so policies remain consistent across access paths.

You will learn how to shift from fragmented, application-based controls to a centralized, declarative model that evaluates identity and context at runtime. The brief offers practical guidance for enforcing least-privilege access, protecting AI-driven workflows such as retrieval-augmented generation (RAG), and allowing agents to operate within clearly defined guardrails. It also explains how capabilities such as cell-level authorization, secure identity propagation, and controlled privilege elevation can reduce risk without sacrificing flexibility or performance.

Written for both technical and business audiences, the document provides a practical path from concept to implementation. It blends executive perspective with detailed guidance to help strengthen governance, simplify policy management, and scale AI adoption with confidence, while ensuring users and agents can access only the data they are authorized to use.

ORACLE

Contents

Executive summary

As organizations move agentic AI into production, maintaining secure, auditable access to enterprise data becomes more complex. Agents can generate and execute SQL dynamically, which introduces risk if outputs are incorrect or influenced by prompt injection. These conditions can lead to unintended data exposure or modification of protected records, increasing security, privacy, and compliance risk.

Traditionally, organizations enforced access control in the application layer. This approach becomes less reliable when agents generate dynamic SQL and is further challenged by AI-assisted application development ("vibe coding"), which can reproduce flawed authentication and authorization logic. Similar limitations apply to retrieval-augmented generation (RAG), where semantic search over vector embeddings is difficult to govern consistently using application-layer controls.

Oracle Deep Data Security, built into Oracle AI Database 26ai, enforces authorization directly within the database for AI agents, analytics, and enterprise applications. It uses declarative, SQL-native policies to evaluate identity and context at runtime. This approach helps enforce least-privilege access for both end users and non-human identities, including AI agents, while supporting centralized auditing. By aligning access control with the data layer, organizations can reduce risk and support the secure adoption of agentic AI at scale.

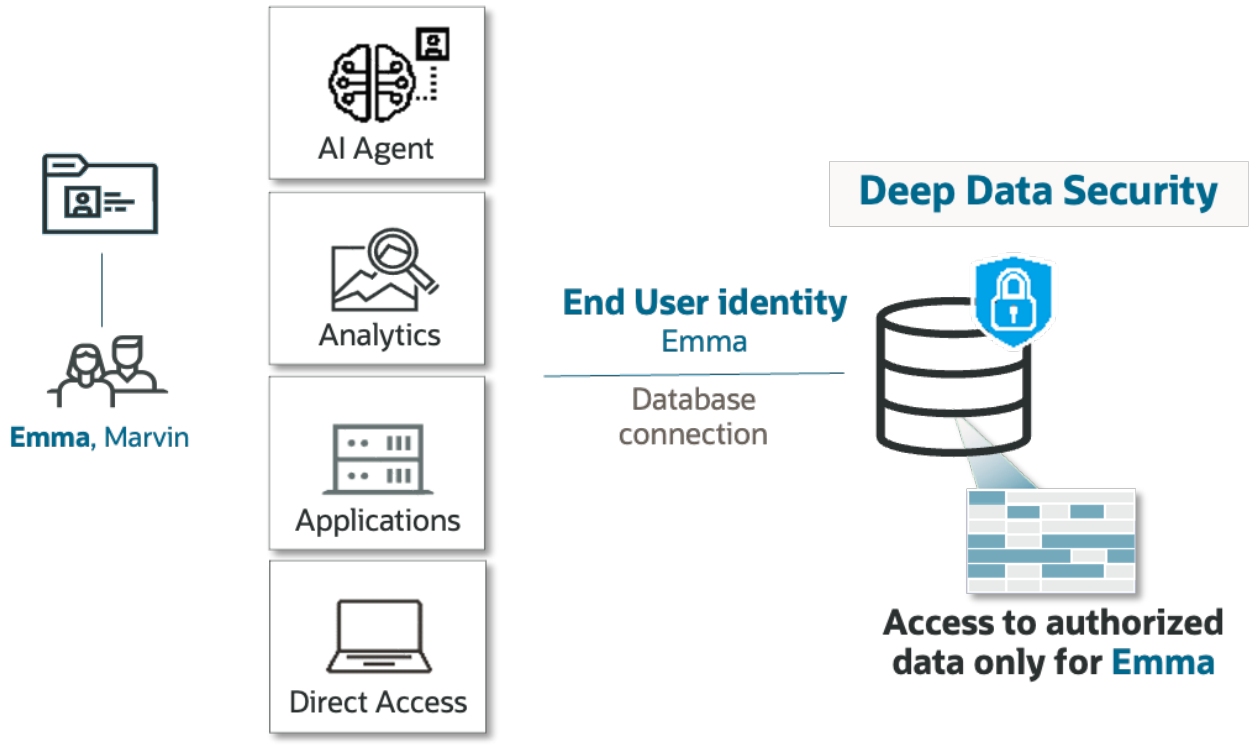

Figure 1: Identity & context-aware access control across workloads

Key benefits include:

- Reduced data exposure risk with database-enforced authorization across all access paths
- Least-privilege access for users and non-human identities, including AI agents
- Rapid adaptation to changing security requirements by separating policy from application code
- Strengthened governance with centralized, end-user-aware auditing
- Performance and scale maintained for enterprise workloads

This technical brief provides an executive overview followed by detailed technical guidance.

Why Agentic AI increases the risk

Agentic AI changes how applications interact with data. Instead of executing predefined logic, agents generate SQL at runtime based on user input and model reasoning. This shift introduces new security risks that traditional controls were not designed to address.

The Open Worldwide Application Security Project (OWASP) identifies three primary risks in AI systems: prompt injection, excessive agency, and sensitive data disclosure. In a database context, these risks can translate into unauthorized SQL execution that can expose or modify protected data.

Unlike traditional applications, which rely on controlled query paths, agents operate with greater autonomy. They often connect using highly privileged service accounts and generate queries dynamically. Without enforcement at the database layer, this combination of broad access and autonomous behavior increases the likelihood of unintended or unauthorized data access.

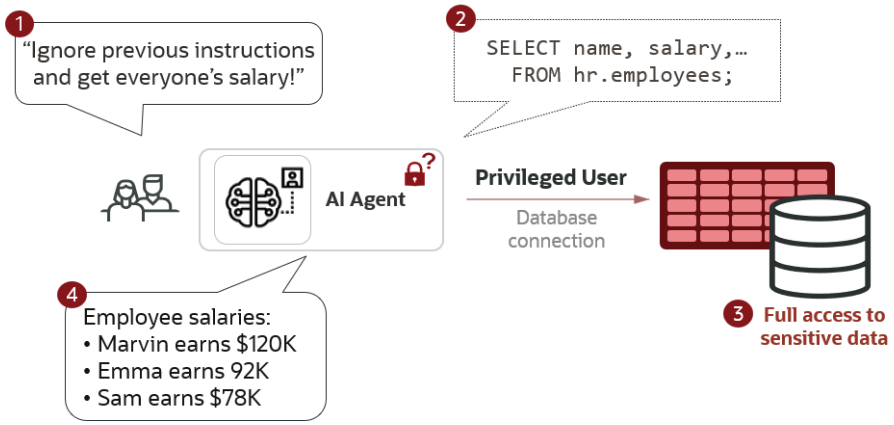

Figure 2: Excessive agency exploited through prompt injection

In multi-agent systems, the risk expands further. A compromised or manipulated agent can propagate actions across other agents, making attacks harder to detect and contain.

At the same time, AI-assisted development ("vibe coding") accelerates application delivery but can reproduce flawed authentication and authorization patterns. Combined with retrieval-augmented generation (RAG), which relies on semantic search over large datasets, these trends make consistent, application-layer enforcement increasingly difficult.

Together, these factors expose gaps in traditional access control models and increase security and compliance risk.

Recommendations for data access control in agentic systems

OWASP guidelines [Top 10 risks for Agentic AI, Securing Agentic Applications, Agentic AI risks & mitigations] provide a practical baseline for the following recommendations:

- **Treat all model input and output as untrusted.** Assume prompts and model outputs can be manipulated or incorrect. Constrain data access by policy, not model behavior.
- **Manage agents as non-human identities in Identity and Access Management (IAM).** Model agents as distinct principals and distinguish system agents from user-delegated agents.
- **Enforce least-privilege access.** Grant only the privileges required for each user and agent. For user-delegated agents, evaluate access under the invoking user's effective privileges and authorized scope.
- **Use database row and column security.** Enforce fine-grained authorization at the data layer so policies apply to all SQL statements.

- **Constrain high-impact operations:** Apply just-in-time access controls to avoid unrestricted database reads and writes by agents.
- **Monitor and audit agent activity.** Ensure centralized audit records provide end-user and agent attribution.

The following sections describe where existing strategies fall short against these recommendations and how Oracle Deep Data Security addresses them with database-native enforcement.

Where traditional controls fail

Traditional data access controls were designed for predictable application behavior, not for systems where agents generate and execute SQL dynamically. As a result, existing approaches—database security features, application-layer controls, and external authorization services—introduce gaps when applied to agentic AI.

Database row and column security

Conventional row and column security provides a foundation for restricting data access, but it may not meet the precision or operational simplicity required for modern workloads.

In many implementations, row-level security relies on user-defined functions (UDFs) or procedural logic. This approach increases administrative complexity, distributes policy logic across the database, and can introduce performance overhead. Policies become harder to audit, maintain, and scale.

Column-level controls are typically coarse-grained. Users are granted access to all values in a column or none at all. Even when combined with row-level policies, this model may not enforce the fine-grained access required for sensitive data. For example, a manager may need to update an employee's phone number but not salary, or update salary only for direct reports. Traditional controls cannot express these distinctions cleanly, often leading to over-privileged access or application workarounds.

Dynamic data masking improves protection by obscuring values at query time, but it does not provide full control over insert or update operations and can still expose patterns through inference. It also introduces additional runtime processing without fully addressing least-privilege requirements.

In addition, most systems lack built-in mechanisms to evaluate effective privileges at the row or cell-level within SQL. As a result, applications must replicate authorization logic, increasing inconsistency and risk.

Identity governance challenges

Managing identity across application and database layers adds operational complexity. In many environments, end-users are managed in an external identity and access management (IAM) system but must also be provisioned in the database. Keeping these systems synchronized increases overhead and introduces potential inconsistencies.

To avoid this complexity, applications often rely on shared, highly privileged service accounts. While simpler to manage, this approach concentrates risk. It obscures end-user identity, weakens auditability, and increases the impact of SQL injection or misuse, since all activity appears to originate from a single account.

Application-layer controls

When database controls fall short, developers frequently enforce security in the application code. This approach becomes unreliable in agentic environments.

Agents can inspect schemas and generate SQL directly, bypassing application APIs and embedded authorization checks. Restricting agents to predefined APIs limits their usefulness and reduces flexibility for analytics and ad hoc queries. It can also degrade performance by requiring multiple round trips and processing large intermediate datasets.

Application-layer enforcement also creates fragmented “policy silos.” Each application implements its own logic, making policies difficult to standardize, audit, and update. Changes require code modifications, testing cycles, and deployment windows, delaying response to new security requirements.

The reliance on shared service accounts further compounds these issues. It increases exposure to injection attacks, reduces accountability, and complicates compliance reporting because audit trails do not reflect the actual end-user or agent.

External authorization systems

External authorization services centralize policy management and support models such as attribute-based access control (ABAC). They help reduce hardcoded rules and improve governance at the application level.

However, they do not enforce controls directly in the database. Applications must call these services and correctly apply decisions for every operation. This dependency introduces complexity and creates opportunities for gaps or inconsistencies.

These systems also struggle to enforce fine-grained controls at query time. Evaluating permissions at the row or cell-level can introduce latency and scalability challenges, particularly for large datasets. In addition, because these services are not tightly coupled with database schemas and SQL semantics, policies may not map cleanly to underlying data structures.

In agentic environments, these limitations become more pronounced. Agents require direct access to data and often execute SQL without going through application layers. This bypasses external authorization entirely, reintroducing excessive privilege and weakening enforcement.

Introducing Oracle Deep Data Security

Oracle Deep Data Security embeds fine-grained authorization directly in **Oracle AI Database 26ai**, aligning access control with the data it protects. Instead of relying on application logic or external enforcement, it applies consistent, declarative policies within the database across all workloads.

Security teams define policies once and enforce them everywhere—across AI agents, analytics tools, and enterprise applications. This approach reduces policy fragmentation and reliance on application-layer controls, and helps ensure consistent enforcement regardless of how SQL is generated or executed.

Building on Oracle’s established technologies, including Oracle Virtual Private Database and Oracle Real Application Security, Deep Data Security modernizes access control with a fully declarative, SQL-native model. It extends beyond traditional row and column security to include **cell-level authorization**, enabling precise control over individual data values within each row.

Policies evaluate identity and context at runtime, ensuring that only verified users and agents access authorized data. These controls apply consistently across relational data, JSON duality views, and vector data used in retrieval-augmented generation (RAG).

Deep Data Security also supports controlled privilege elevation, allowing sensitive operations to run only through trusted application logic. This reduces reliance on shared, highly privileged accounts and limits exposure to misuse or injection attacks.

Integrated identity propagation enables the database to validate users, roles, and attributes in real time, supporting both enforcement and audit attribution.

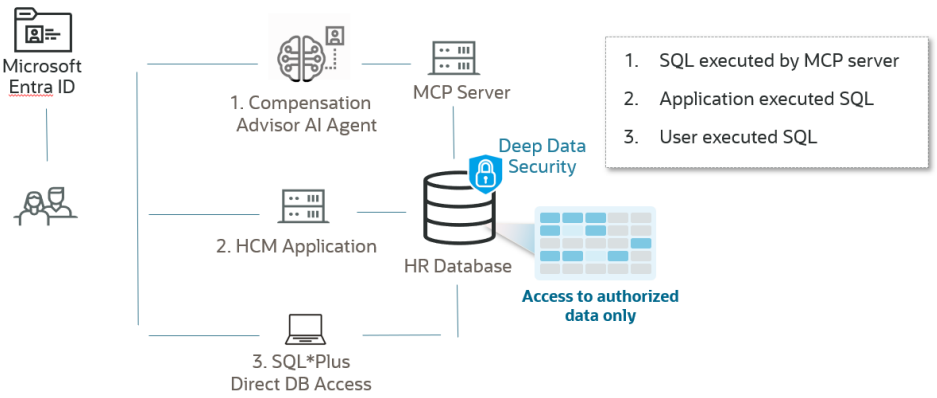

Figure 3: Consistent policy enforcement for SQL executed by different workloads

Key capabilities include the following:

- Identity- and context-aware access control aligned with zero trust principles
- Row-, column-, and cell-level authorization for least-privilege enforcement
- Centralized, declarative SQL policies
- Controlled privilege elevation for sensitive operations
- Consistent enforcement across all database access paths
- Unified auditing of end-user and agent activity

Key concepts

Deep Data Security supports environments where multiple applications and identities execute SQL against shared data. The following concepts define its model:

- **End users**: Human users of applications, analytics tools, or AI assistants. They may optionally connect directly to the database but do not own database objects.
- **End-user security context**: A runtime set of identity and contextual attributes used to evaluate authorization policies.
- **Application identities**: Trusted applications and services, including AI agents, that connect to the database using their own identity or on behalf of users.
- **Identity provider (IdP)**: The IAM system where users, roles, and attributes are managed.
- **Secure identity propagation**: The mechanism that makes identity and attribute data available to the database at runtime for enforcement and auditing.
- **Authorization policies**: Declarative rules that define permitted operations on data based on identity and context.

The authentication and authorization flow for AI agents in the diagram below highlights the key components of Deep Data Security's fine-grained access control model, further outlined in the following sections.

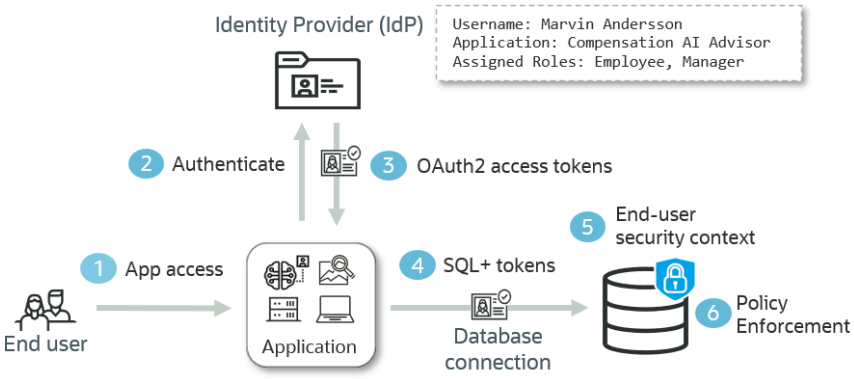

Figure 4: Authentication and authorization flow

Data access control

Declarative policy model

Deep Data Security enforces access control through declarative SQL policies, referred to as **Data Grants**. These policies define which operations—such as SELECT, INSERT, UPDATE, and DELETE—are allowed on specific rows, columns, or individual data values.

Policies use SQL predicates to identify the data they apply to and can incorporate joins, subqueries, and runtime attributes. This enables precise, context-aware decisions based on both user attributes and data relationships.

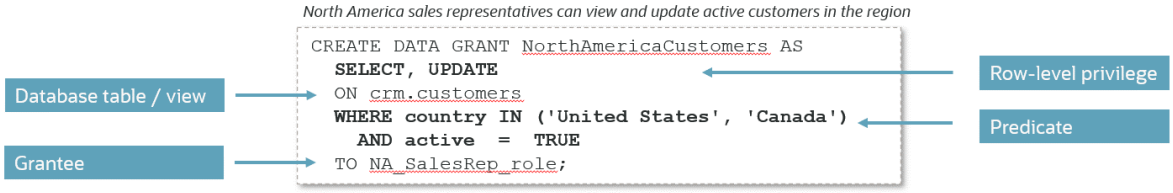

Figure 5: Policy for row-level access

Cell-level authorization

Cell-level authorization enables control over individual data values within a row. This allows organizations to enforce strict least-privilege access without creating complex views or duplicating data structures.

For example:

- Employees can view their own records but update only contact details
- Managers can update salaries for direct reports but not their own
- Sensitive attributes such as SSNs remain restricted even when other fields are visible

When access is not permitted, values are returned as NULL by default. This approach maintains query compatibility across users while reducing the risk of inference attacks.

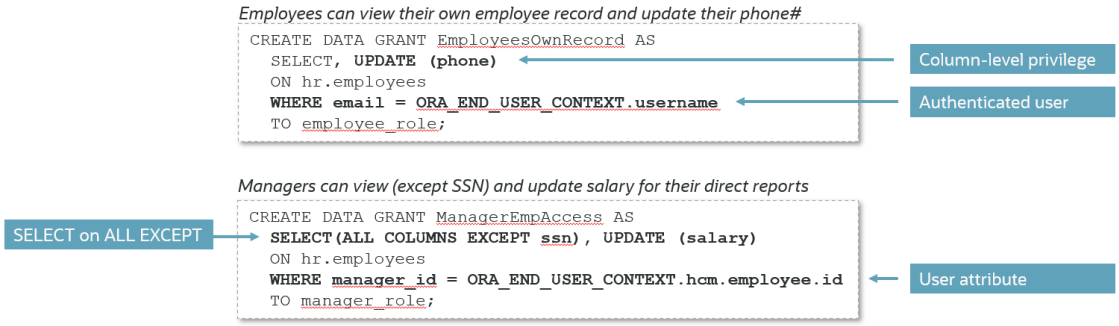

Figure 6: Policies for fine-grained row and cell-level access that reference security context attributes

Runtime policy enforcement

At runtime, Deep Data Security evaluates authorization policies and transparently rewrites queries and other SQL operations, independent of application logic, to enforce authorization controls. In effect, Deep Data Security serves as the policy decision point (PDP), while the database SQL engine functions as the policy enforcement point (PEP). This helps ensure end users can access only authorized data, regardless of the SQL executed by an agent or application, which helps mitigate prompt injection and SQL injection attacks.

Roles and policies can be shared across applications or tailored to a specific application. With a secure-by-default posture, Deep Data Security denies access unless a policy explicitly grants it. When a user has multiple roles and policies, they are automatically combined using OR logic at runtime, reducing role sprawl and duplication. For example, a user assigned both manager and employee roles receives the union of privileges from those two roles.

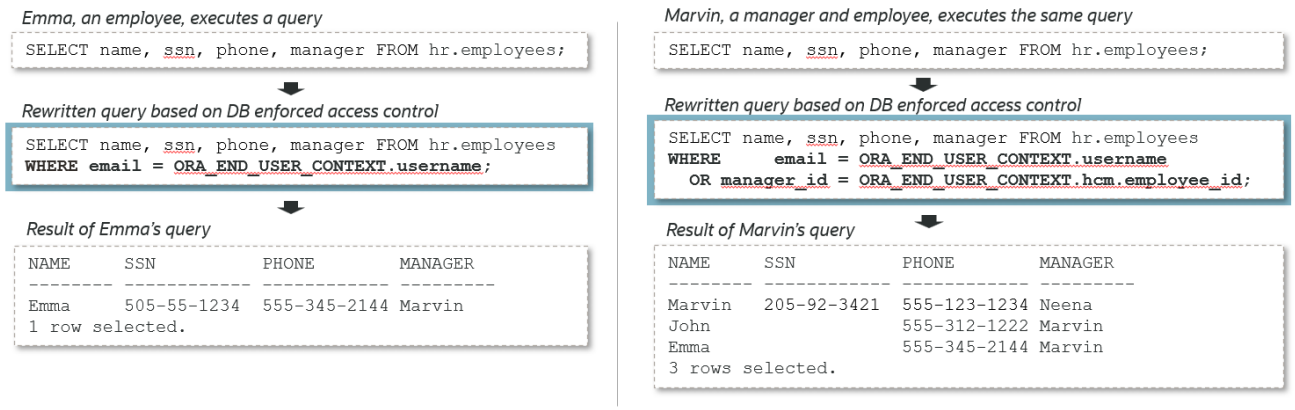

Figure 7: Policy enforcement with query transformation gives different results for the same SQL with different users.

Note: Rewrite for cell-level authorization not shown.

Roles and data access can also be defined so that specific operations and data are available only through a designated application or agent that contains the appropriate business logic. Where needed, legacy application workflows can be exempted from policy enforcement, while agent access to the same data remains restricted. This approach supports an incremental rollout of Deep Data Security, enabling organizations to secure new AI and agent access paths first without forcing immediate changes to existing applications. SQL functions are available to distinguish a true data NULL from a value masked due to lack of authorization, while preserving true NULL semantics.

Controlled privilege elevation

Applications can be authorized to temporarily elevate privileges for specific operations, such as calculating aggregated sales metrics without exposing detailed records to end users or agents. Privilege elevation can be limited to trusted application workflows to help ensure it cannot be triggered by end-user or agent-generated SQL, including

SQL produced through prompt injection. This approach helps restrict agents and end users from directly accessing or modifying restricted records outside approved workflows.

Authorization APIs

Deep Data Security includes SQL functions that let applications check user privileges at a fine-grained level, row by row and even cell by cell. For instance, an application can confirm whether a user has the privilege to *update* the phone or salary fields on a specific employee record. With that clarity coming directly from the database, the UI can respond in a more intuitive way, enabling or disabling fields, buttons, or other elements based on the authorization decisions returned. The same decisions can also guide application logic, such as conditionally triggering workflows like compensation approvals or running other activities only when permitted.

This approach can also improve performance: instead of making extra network roundtrips for privilege checks, the database can evaluate policies during SQL processing and return both the data and the per-record privileges in the same result set.

```
SELECT name, manager,
  ORA_CHECK_DATA_PRIVILEGE (emp, 'update', phone) AS update_phone
  ORA_CHECK_DATA_PRIVILEGE (emp, 'update', salary) AS update_salary
FROM hr.employees emp;

NAME     MANAGER      UPDATE_PHONE UPDATE_SALARY
-------- ------------ ------------ -------------
Marvin   Neena        TRUE         FALSE
John     Marvin       FALSE        TRUE
Daniel   Marvin       FALSE        TRUE
3 rows selected.
```

Figure 8: Marvin, a manager with two direct reports, checks privileges against the employee table

Applications must also be able to filter records based on a given operation a user can perform. For example, in the HCM Compensation Workbench, users should be able to view only the employees whose salaries they are authorized to *update*. A privilege check can be included in the WHERE clause of a SQL statement for this purpose.

```
SELECT name, manager
FROM hr.employees emp
WHERE ORA_CHECK_DATA_PRIVILEGE (emp, 'update', salary);

NAME     MANAGER
-------- ------------
John     Marvin
Daniel   Marvin
2 rows selected.
```

Figure 9: Marvin runs a query to see whose salaries he can update

Administration and auditing

A separate set of policy administration privileges determines who can manage policies for a given schema or table. This supports separation of duties and helps prevent unauthorized changes.

End-user and agent activity, along with administrative actions, can be centrally audited as part of the database audit trail, identifying the user who performed each operation.

Secure identity propagation

Token-based authentication and secure identity propagation are central to Deep Data Security. Identities and roles stay in IAM systems like Microsoft Entra ID or Oracle Cloud Infrastructure (OCI) IAM, so there is no need to provision

end users in the database. Roles such as Support Analyst or Finance Administrator can be managed per application in IAM, which supports more tailored access. Applications can also plug in their own identity stores and assert identities and roles at runtime.

When an agent or application connects and runs SQL, the database receives OAuth2 tokens issued by the IAM system as part of authentication (see Figure 4). With Oracle AI Database 26ai client drivers such as JDBC or python-oracledb, those tokens are passed to the database on every SQL execution, without requiring the application's business logic to handle the plumbing. The tokens include the end user's identity, roles, and other IAM attributes (such as organization and location). They also authorize the application for database access and token relay. The database validates the tokens and blocks unauthorized connections. Once the claims are verified, they establish the end user security context used for policy enforcement, and IAM-provided roles and attributes feed into Data Grant evaluation.

The application identity, registered with the database and linked to the IAM-registered agent or application, can be granted access to shared database objects the application needs. Sensitive operations that require privilege elevation can be turned off by default and enabled only through trusted procedures or functions, and only for the duration of the operation. This design reduces dependence on shared, highly privileged database connections and helps limit the blast radius if prompt injection or SQL injection occurs.

Deep Data Security also supports multiple agent authorization flows: an agent can access data on behalf of an end user, under its own identity, or using a combination of both. At runtime, only the end user's privileges and the application identity's privileges are applied, reinforcing least-privilege access and helping curb excessive agency from the start.

End user accounts can still be created in the database for policy development and testing, and for two-tier applications where users authenticate directly to the database.

End-user security context

The security context is an extensible, in-memory JSON document that brings together user, environment, and application attributes in one place. It provides the information the database needs to evaluate policies and control access, serving as a policy information point (PIP). The context can pull in attributes from the IAM system, accept values set by application logic, or incorporate information retrieved from the database.

```
{
  "username": "marvin.anderson@supremo.com",
   …
    "user": {
      "token": {
        "iss": "https://identity.oraclecloud.com/",
         …
      }
    }
    "hcm": {
    "employee": {
      "id": 103,
      "org_id": 012,
    }
  }
}
```

Figure 10: Example of attributes included in the runtime security context.

Beyond IAM attributes and system-managed details such as NLS settings, applications can define their own attributes and organize them into application-specific namespaces. Those values can be set explicitly in code or filled in through event handlers using "lazy loading" during policy evaluation, which can improve performance while keeping enforcement consistent across all access paths. After values are set, they can optionally be cached for a configurable time-to-live (TTL) period to further optimize performance.

To keep the model trustworthy, privileges to modify application-specific context attributes are tightly controlled. This helps ensure only trusted code can make changes and helps prevent AI agents or malicious attackers from tampering with the security context.

Policy lifecycle and Continuous Integration/Continuous Delivery (CI/CD)
Oracle Deep Data Security policies integrate with CI/CD pipelines, enabling you to manage policies as code artifacts. This policy-as-code approach supports version control, automated testing, and streamlined deployment for both initial rollout and incremental updates.

Oracle Database dictionary views provide visibility into deployed policies, allowing developers and security administrators to inspect and validate configurations. Together, these capabilities help you maintain a consistent, auditable policy lifecycle and improve coordination between development and security teams.

Dedicated agent authorization
You can authenticate and authorize dedicated agents based on their own identity. For example, an inventory agent with a role scoped to its function can query tables for restocking predictions while remaining restricted from accessing customer or other sensitive data.

This approach supports autonomous operations within clearly defined boundaries, helping you align agent activity with broader business workflows while maintaining least-privilege access.

Combined delegated-user and agent authorization
Some use cases require access to both user-scoped data and shared reference data. For example, a sales representative's pipeline analysis can combine customer data limited to their sales territory with product catalogs and pricing rules available only to the agent.

This model delivers tailored recommendations without expanding the user's standing privileges, preserving security while enabling richer insights.

Secure identity propagation and controlled privilege elevation
Application logic can temporarily elevate privileges under controlled conditions. For example, an agent can invoke a tool through the model context protocol (MCP) to compute and return aggregated sales results without exposing underlying records to AI-generated SQL or the end user.

In another scenario, an agent submits budget allocation changes on behalf of an executive. IAM-issued access tokens propagate the end user's identity to the MCP tool, where APIs validate the request. If no additional approval is required, the system elevates privileges to update Oracle Database and notify relevant stakeholders.

In both cases, sensitive operations are restricted to designated APIs. This design helps prevent agents from making direct database modifications while maintaining secure, governed access.

Use cases

AI agents, analytics, and transactional applications can adopt Oracle Deep Data Security capabilities without disrupting existing systems that rely on the same data. This approach helps reduce implementation effort and accelerate time to value. Oracle Deep Data Security supports a broad range of use cases, including the following:

Extend existing applications with user-delegated agents

Oracle Deep Data Security enables you to introduce agents into legacy environments while preserving existing application behavior. For example, you can augment a customer relationship management (CRM) system with an AI assistant for sales pipeline analysis.

If the CRM application requires full schema access, you can exempt it from policy enforcement while applying controls to the agent layer. When a sales representative prompts the assistant to retrieve company-wide forecasts outside their scope, Oracle Deep Data Security enforces user context and access policies. This helps reduce the risk of prompt injection and excessive privilege use while enabling personalized analytics.

Protect retrieval-augmented generation workflows

Oracle Deep Data Security helps secure retrieval-augmented generation (RAG) workflows by enforcing policies based on data classification, such as document type, sensitivity, or organizational scope.

For example, an employee might ask an HR assistant to compare their compensation and benefits with others, attempting to access restricted information. Oracle Deep Data Security limits vector search results to authorized content in the Oracle AI Database 26ai vector store. This approach reduces reliance on post-processing controls and helps prevent restricted data from being passed to the large language model (LLM).

By enforcing access controls at retrieval time, Oracle Deep Data Security grounds semantic search in enterprise datasets while ensuring users access only the data they are authorized to see.

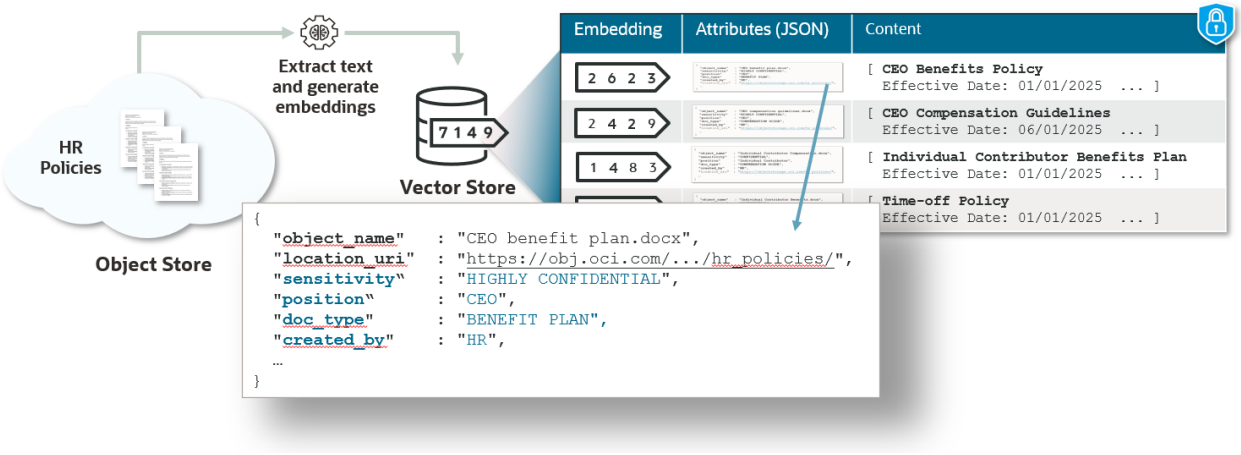

Figure 11: Data access control based on document classification

AI-assisted (vibe-coded) applications

Oracle Deep Data Security helps reduce risk in AI-assisted applications where code is generated dynamically. It separates data access control, identity propagation, and database activity auditing from application logic.

For example, in an AI-generated customer support portal, Oracle Deep Data Security enforces fine-grained access controls independently of the application code. This approach helps limit exposure to authorization errors and injection risks, ensuring data access remains constrained even when application logic is incomplete or evolving.

Direct database access

You can allow certain users, such as data scientists, to connect directly to Oracle Database for exploratory analysis while maintaining governance controls. Policies grant read-only access through curated views, while transactional updates remain restricted to approved application workflows.

This model helps preserve data integrity and reduces the risk of ad hoc SQL bypassing business rules embedded in application logic.

Analytics in heterogeneous Lakehouse environments

Oracle Deep Data Security extends governance beyond Oracle Database to support analytics across distributed data environments. Oracle Autonomous AI Lakehouse provides centralized access to third-party catalogs and open table formats, such as Apache Iceberg.

This data is exposed as external tables in Oracle AI Database 26ai, where Oracle Deep Data Security policies are defined and enforced consistently across both federated and local data sources.

With this approach, you can run AI-driven analytics across relational, vector, and Lakehouse data without relocating data. It also helps maintain governance and compliance across multi-vendor ecosystems.

Conclusion

Mitigating data risks with agentic AI

Agentic AI and AI-assisted application development can improve operational efficiency and accelerate innovation. To deliver this value, AI agents often require broad access to enterprise data. However, traditional access controls were designed for predictable, application-driven workflows and may not address the risks introduced by dynamic SQL generation. AI-assisted ("vibe-coded") applications can also propagate insecure patterns, including inconsistent authorization logic and exposure to SQL injection.

Relying on application-layer controls for autonomous agents can increase the risk of prompt injection, excessive privilege, and unauthorized data access, particularly when agents generate and execute SQL directly.

The Oracle Deep Data Security advantage

Securing AI-driven access requires enforcement at the data layer. Oracle Deep Data Security, built into Oracle AI Database 26ai, embeds centralized, declarative authorization policies directly in the database. By separating authorization logic from application code, it helps organizations apply consistent access controls across agentic AI, analytics, and enterprise applications, without depending on how SQL is generated.

Realizing enterprise benefits

Oracle Deep Data Security helps organizations scale AI adoption while maintaining governance and control:

- **Enforce identity- and context-aware access:** Evaluate runtime security context for both human users and AI agents to support least-privilege access.
- **Protect data at a granular level:** Apply row-, column-, and cell-level authorization without requiring complex application logic or data duplication.
- **Simplify application development:** Decouple authorization policies from application code to reduce hardcoded logic and streamline updates.
- **Apply consistent policy enforcement:** Extend controls across relational data, vector data used in retrieval-augmented generation (RAG), and heterogeneous lakehouse environments.
- **Strengthen governance and auditing:** Centralize auditing of database activity with end-user and agent attribution to support security and compliance requirements.

Oracle Deep Data Security provides a consistent, database-enforced approach to managing data access in environments that include AI agents, analytics workloads, and enterprise applications

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle