ORACLE

Deep Data Security FAQ

Agentic AI systems increasingly interact with production databases to read and write data. Excessive agency, prompt injection, and other security risks can allow guardrails to be bypassed and expose data a user is not authorized to access.

Oracle Deep Data Security addresses this by enforcing fine-grained, database-layer authorization for agentic AI, analytics, and enterprise applications. Built into Oracle AI Database 26ai, Deep Data Security applies data controls based on user identity and runtime context. Declarative SQL policies let developers enforce row, column, and cell level controls that limit end users to only the data they are authorized to see, even if the application or agentic AI layer is compromised or makes an error.

Last updated: 1 May 2026

Frequently Asked Questions

Overview

What is Oracle AI Database 26ai Deep Data Security?

Oracle AI Database 26ai Deep Data Security, or Deep Sec, provides a declarative data access control framework in the database. It enables security teams and developers to define fine-grained authorization policies that are enforced in the database based on verified user identity and runtime context, reducing dependence on application logic and AI guardrails to prevent unauthorized data access.

Why is this especially important for agentic AI?

Agentic AI systems can dynamically generate and execute SQL. Prompt injection and excessive agency can cause agents to expose or alter sensitive data, especially when applications or agents connect through highly privileged service accounts. Deep Data Security helps mitigate AI risks by enforcing fine-grained data authorization in the database, so access is constrained by policy, not by model behavior. It mitigates excessive agency risk by confining agents' SQL execution to end-user security context (roles, attributes).

Is Deep Data Security only for agentic AI use cases?

No. Deep Data Security is designed for environments where multiple applications including agentic AI, analytics, transactional applications, RAG workflows, and extending legacy applications with agents, issue SQL against the same database objects. Use cases include:

> **Least-privilege access for users and agents:** Enforce row, column, and cell access on SQL executed by the agent using the end-user's or agent's privileges.

Authorization for AI-generated applications: Enforce access in the database independently of AI-generated application code that may contain authorization flaws.

Extend legacy applications with agents: Enforce fine-grained access for agents using shared schemas, while exempting legacy systems to avoid changes.

Secure retrieval-augmented generation: Enforce classification-based policies on AI vector search so that only authorized content is returned to agents.

Is Deep Data Security for new applications or existing applications?

Deep Data Security supports both new and existing applications. However, the initial release is primarily targeted toward Agentic AI applications.

Deep Data Security enables AI agents to be added to existing applications while keeping existing application behavior unchanged. For example, an existing CRM system could be augmented with an AI assistant for sales pipeline analysis. The CRM application can be exempted from Deep Data Security policy enforcement, as it was designed to require full schema access. However, when a sales representative directs the AI agent to retrieve company-wide forecasts, Deep Data Security ensures that the AI agent operates only within the end user's access rights and context, mitigating excessive agency while enabling personalized analytics.

Do Oracle Deep Data Security policies affect query execution plans?

Oracle Deep Data Security integrates with the database optimizer by augmenting queries with additional security predicates. These predicates are visible to the optimizer, allowing it to select efficient execution plans. This behaves similarly to queries with WHERE clauses, so standard optimization techniques such as indexing remain applicable.

What performance impact can I expect from Deep Data Security?

Deep Data Security performance impact is negligible. The primary factor that can affect performance is how SQL predicates used in policies are defined. These predicates are evaluated as part of the SQL WHERE clause, so standard Oracle SQL best practices apply. For example, predicates should be written to take advantage of indexes and, where applicable, partitioning (such as enabling partition pruning), and avoid patterns that prevent the optimizer from using them effectively.

Security model: identity, context, and least privilege

How are end-user identity, roles, and attributes propagated to the database for enforcement?

When agents or applications connect and execute SQL, the database receives OAuth 2.0 tokens issued by the IAM system and passed via client drivers (examples include JDBC and Python) on each SQL execution. The database validates the token and uses verified claims (identity, roles, attributes) to establish the end-user security context used for policy enforcement and audit attribution.

What is the "end-user security context"?

The end-user security context is a session-level object managed by the database that contains the end user's identity, roles, and attributes used to evaluate access control and enforce data security policies. It is created automatically for each request and governs what data the user or agent can access at runtime.

This context includes attributes from multiple sources, such as the identity provider and application logic. These attributes are evaluated by the database to enforce fine-grained access control policies.

Attribute values can be set directly by the application or resolved through event-handler functions that are invoked on first access (lazy loading).

Permissions to modify application-specific attributes are enforced by the database through controlled mechanisms such as context schemas and associated handler functions. Only authorized PL/SQL handler functions can set or update these attributes, preventing arbitrary modification by applications.

Can Deep Data Security help eliminate shared high-privileged service accounts for agents and apps?

Yes. Oracle Deep Data Security reduces or eliminates the need for shared, highly privileged database accounts by enabling identity-aware access control at runtime.

Applications and agents operate using the end user's identity and context (including roles and attributes), which are propagated to the database and enforced through policies. This requires that user-specific access rights (for example, roles and data grants) are already defined in the system. Oracle Deep Data Security ensures these privileges are consistently enforced for every query, regardless of how it is generated or executed.

The underlying database connection can run with minimal privileges (for example, CREATE SESSION and CREATE END USER SECURITY CONTEXT), and does not require direct object-level privileges on tables or views. Access is determined entirely by the end-user security context and associated policies.

It enforces agent authorization flows (on behalf of end users, by agent identity, or combined) where only the end user's and application identity privileges take effect at runtime, enforcing least privilege and mitigating excessive agency.

How does Oracle Deep Data Security work with connection pooling and shared database accounts?

Enhanced client drivers transparently propagate end-user identity, roles, and attributes to the database for every database operation, even when applications use connection pools with shared database accounts. This enables enforcement of least-privilege access based on end-user context rather than relying on highly privileged shared accounts. Support is available for drivers such as JDBC and Python, with additional support planned for other language drivers.

Agentic application development model

What are agent authorization flows?

Deep Data Security does not impose any restrictions on authorization flows for agentic applications. Some of the authorization flows are:

1. End users/agents/applications access data
2. Agents acting on-behalf-of end users
3. End users operating in the context of applications/agents

There are two primary ways an end user can access data through applications:

1. Using their own identity: The application is either 2-tier where the user is authenticated directly to the database, or 3-tier where the end user accesses the database by consenting to provide an on-behalf-of authorization assertion to the application.
2. Using the application identity: The database either trusts the application to pass the end-user identity attributes or requires the database to assert that the end user is a legitimate user of the application. This has two variations: (a) the middle tier hosts a single application, and the connection-pool identity represents the application, and (b) the middle tier hosts multiple agentic applications, where each agentic application has a distinct set of database privileges.

For the initial release, Oracle Deep Data Security focuses on flows 1 and 2(b). Flow 2(a) is planned to be supported in the next release.

Is using an identity provider required for developing agentic applications using Deep Data Security?

Oracle Deep Data Security is primarily designed for cloud-based and agentic applications that integrate with an identity provider, such as Microsoft Entra ID or Oracle Cloud Infrastructure (OCI) IAM, to propagate end-user identity, roles, and attributes to the database.

However, using an identity provider is not strictly required in all scenarios. Oracle Deep Data Security also supports locally managed end users, where identity and data roles are defined and resolved entirely within the database. In this model, applications can pass an end-user security context using a user name, or users can connect directly using database credentials, without relying on the identity provider.

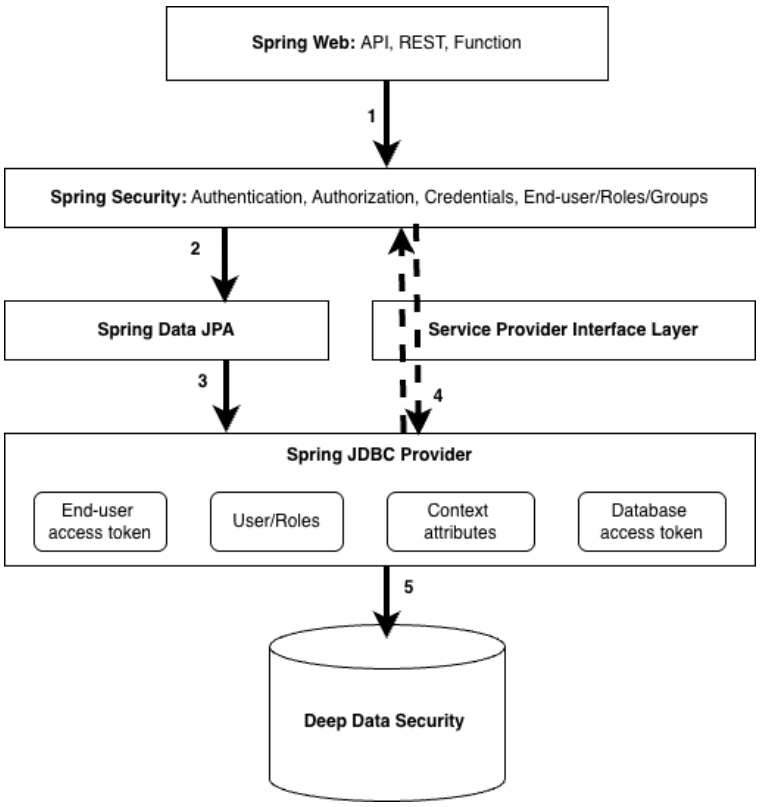
Locally managed end users are typically suited for simpler architectures, standalone applications, or development and test environments, while identity provider-based integration is recommended for modern multi-tier and agentic application patterns.

Do I need to develop application code to uptake Deep Data Security?

Oracle drew on its experience with Oracle Real Application Security (RAS) to develop Deep Data Security. To adopt RAS, application code must access the database connection object to create and attach RAS sessions. Because connection objects are often abstracted behind Java persistence layers (for example, Hibernate) and connection pools (for example, HikariCP), adopting RAS typically require extending these modules.

For Deep Data Security, Oracle removed all client-side APIs; instead, it automatically manages the lifecycle of the end-user security context on the database server. In addition, no changes are required to the data access/persistence layer, because SQL driver callbacks capture the security context directly from the application's framework-level security context. Finally, Deep Data Security uses a configuration-driven Service Provider Interface (SPI) model to integrate with application and agent frameworks. Instead of writing custom code to manage and propagate security context, developers configure the framework and the driver automatically handles identity propagation. This approach reduces development effort and accelerates time-to-value. In public repositories, Oracle will provide out-of-the-box, open-source Service Provider Interface (SPI) implementations, and sample code for popular application and agent development frameworks.

The following example uses the Spring Boot Java application development framework.



When a user calls the application API (1), Spring Security authenticates the user via Entra ID or OCI IAM, and stores the user's identity, token, roles, groups, and claims in the Spring security context (2). Spring Data JPA continues to handle database operations as usual (3). During every JDBC call to the database, the Oracle JDBC SPI callback function is executed. The JDBC provider for the SPI reads the current Spring Security context (4) and passes the required end-user security information to the Oracle AI Database 26ai (5). Deep Data Security then uses that context to enforce data access policies inside the database without requiring code changes in JPA (such as Hibernate) or in the connection pool layer (such as HikariCP) within a standard configuration-based setup.

Can Model Context Protocol (MCP) servers support Deep Data Security?

MCP servers can expose tools to access Oracle databases (for example, executing SQL or accessing the database internally).

- Custom MCP servers: Sample code for Spring Boot (Java) and Django, FastAPI, and Flask (Python) will be provided in the documentation.
- Oracle-hosted SQL MCP servers (SQLcl, AI DB MCP, DB Tool MCP): We are working with those teams to adopt Deep Data Security.

Which MCP clients can interact with MCP server with Deep Data Security support?

Any MCP client can work with an MCP server that supports Oracle Deep Data Security. The security is handled on the server side (in the database access tools), not in the client. The MCP server connects to the database using Oracle drivers (like JDBC). To use Deep Data Security, the server must use OAuth to authenticate users and pass their identity to the database. If the server is set up this way, any MCP-compatible client will work.

Which AI agent development frameworks are supported for Deep Data Security?

There are primarily two types of AI agent development SDKs: generic AI libraries such as LangChain, LangGraph, and LlamaIndex; and SDKs with multi-agent development support such as Microsoft AutoGen, CrewAI, Google Vertex AI Agent Builder, and the OpenAI Agents SDK.

For these frameworks, using Deep Data Security requires adopting Oracle's new SQL client drivers and leveraging the Deep Data Security SPI or sample code. For example, as Python libraries such as FastAPI are widely used for AI agent development, the primary integration work involves using the Python client driver with FastAPI.

Fine-grained authorization and runtime enforcement

How does Deep Data Security protect data when AI agents dynamically execute SQL?

Oracle Deep Data Security enforces authorization at the database layer by binding every SQL execution to the end user's identity, roles, and attributes at runtime. When an application or agent submits a query, the database establishes an end-user security context and evaluates data access policies (data grants) as part of query execution. These policies are applied directly to the SQL, dynamically filtering rows and restricting column or cell values based on the user's privileges. As a result, even if an AI agent generates arbitrary or malicious SQL, it can only access data that the end user is explicitly authorized to see, helping effectively mitigate excessive agency, prompt injection, and SQL injection.

This model removes reliance on application logic or AI guardrails. Authorization is enforced consistently by the database itself, across all queries and access paths, ensuring that unauthorized data is never returned—even if the query is incorrect, injected, or generated dynamically.

How fine-grained are the controls (row, column, cell)? What happens when access is denied?

Oracle Deep Data Security provides fine-grained controls at the row, column, and cell level. This means access can be restricted not just to specific rows or columns, but to individual values within a row.

At a high level:

- Queries return only the data the user is authorized to access.

- If a user is not authorized to see a specific value, it is returned as NULL by default.
- The same SQL query can safely run for different users, with results automatically filtered based on their authorization.

For data modification:

- UPDATE operations require permission for all values being changed. If any value is not allowed, the update is not done.
- INSERT operations require permission for the data being inserted. If the data does not meet access rules, the operation fails with an error.
- DELETE operations are controlled at the row level and require delete permission on the target rows.

In all cases, access decisions are enforced by the database at runtime, ensuring users and agents can only read or modify data they are authorized to access.

What happens when a security check fails, and how does Oracle Deep Data Security distinguish between missing and unauthorized data?

Oracle Deep Data Security follows standard database behavior: queries and DML operations continue to execute, but unauthorized data is not exposed. If no data exists that matches a query, no rows are returned. If a user is not authorized to access a specific value, that value is returned as NULL by default.

To distinguish between an actual NULL and a value hidden due to lack of authorization, applications can use the ORA_IS_COLUMN_AUTHORIZED SQL function. This allows applications to handle both cases appropriately without changing query logic.

How can applications check privileges to drive UI actions (enable/disable buttons, fields, workflows)?

Deep Data Security provides SQL functions—for example, ORA_CHECK_DATA_PRIVILEGE, ORA_IS_COLUMN_AUTHORIZED — that enable applications to check privileges row by row and at cell/column level. This enables apps to dynamically tailor UI and workflows and can reduce network roundtrips by returning both data and privilege decisions with the query result.

Can queries be filtered based on what a user can do (not just what they can see)?

Yes. Privilege checks can be included in the WHERE clause, so applications can return only the records for which a given operation is allowed. For example, show only employees whose salaries a manager is authorized to update.

Custom privileges that represent application-level operation will be supported in future releases.

RAG, analytics, and heterogeneous data

Can Deep Data Security secure retrieval-augmented generation (RAG) and vector search?

Yes. Deep Data Security can help secure RAG workflows by enforcing policies based on data classifications (document type, sensitivity, organization) so retrieval is limited to authorized document segments in the vector store and restricted content is prevented from being passed to the Large Language Model (LLM).

Does Deep Data Security support modern data types and lakehouse / federated data?

Yes. Policies can be applied across data types including relational data, JSON Duality Views, and vector embeddings used in RAG. Deep Data Security also supports centralized enforcement across federated and local data where third-party catalogs (for example, Snowflake and Databricks) and open table formats (for example, Apache Iceberg) surface as external tables in Oracle AI Database and are subject to the same policies.

Governance, auditability, and lifecycle

How does Oracle Deep Data Security support governance—auditing and managing policy changes over time?

Activities can be centrally audited in the database audit trail with end-user or administrator attribution. Policies are designed to support CI/CD pipelines through a policy-as-code model, enabling version control, automated testing/deployment, and inspection via database dictionary views—supporting an auditable lifecycle for policy changes.

Is there a user interface for authoring and managing Oracle Deep Data Security policies?

Policies are currently authored using SQL, which provides flexibility and supports automation through CI/CD workflows. Integration with user interfaces such as Oracle APEX and Oracle Data Safe is planned to simplify policy authoring and management.

Comparisons and coexistence with existing Oracle controls

How is Oracle Deep Data Security different from Virtual Private Database (VPD)?

Virtual Private Database requires applications to implement custom authorization logic—typically through PL/SQL functions—which can introduce inconsistencies and increase maintenance overhead at scale. Oracle Deep Data Security provides a declarative, centralized model where policies are defined in SQL and enforced uniformly in the database, independent of application logic.

Can Oracle Deep Data Security replace VPD?

Oracle Deep Data Security provides a modern, declarative alternative to VPD and can address many of the same use cases with a simpler model. Migration from VPD is not automatic and may require some development effort. Future enhancements, including SQL Macros integration, are expected to further simplify transitions.

How is Oracle Deep Data Security different from Real Application Security (RAS), and what are its advantages?

Oracle Deep Data Security modernizes the capabilities introduced with Oracle Virtual Private Database (VPD) and Oracle Real Application Security (RAS) for agentic AI, analytics, and enterprise application workloads. It replaces procedural PL/SQL and API-driven controls with declarative policies defined in SQL, simplifying adoption and reducing reliance on application changes.

With Deep Data Security, session and security context management are handled entirely on the database server and are automatically managed. In contrast, RAS requires application code to access database connection objects to create and attach sessions. Because connection objects are often abstracted behind Java persistence layers such as Hibernate and connection pools such as HikariCP, adopting RAS can require extending these components.

With Deep Data Security, the drivers are extended to provide native end-user security context management through a Service Provider Interface (SPI). Context propagation is delivered via the SPI to remove dependencies on data-access-layer components (for example, Hibernate) and connection pools (for example, HikariCP). It also supports a broader range of use cases and extends beyond row- and column-level controls with fine-grained authorization at the cell level.

Are VPD and RAS being desupported?

No. VPD and RAS will continue to be fully supported; Deep Data Security offers a new, modern capability that complements the existing toolkit.

Can Deep Data Security work alongside other Oracle security controls (for example Label Security, Database Vault)?

Yes. Oracle Deep Data Security complements existing database security features and does not replace them. Controls such as Database Vault, Data Redaction, and Label Security continue to play important roles in protecting data.

Oracle Deep Data Security adds fine-grained, runtime authorization based on identity and context, particularly for agentic AI and modern application architectures. For example, Label Security can still be used to control which rows are returned based upon labels, while Database Vault can restrict privileged user access, enforce trusted paths, and limit the use of sensitive operations such as TRUNCATE or DROP TABLE.

Can Data Redaction be used for identity-aware access control?

Data Redaction is designed for masking sensitive data but may be vulnerable to inference risks when used alone. Oracle Deep Data Security incorporates dynamic masking as part of a broader access control model, enabling identity- and context-aware authorization with fine-grained controls similar to, but more advanced than, column-level VPD.

In the initial release, unauthorized column or cell values are returned as NULL by default. Applications can use SQL functions such as ORA_IS_COLUMN_AUTHORIZED to explicitly determine whether a value is NULL due to lack of authorization and adjust query logic or presentation accordingly.

Do customers still need tablespace encryption?

Yes, tablespace encryption solves a different problem. Oracle Deep Data Security enforces authorization based on end-user identity to determine whether a column, row, or cell value should be returned. Tablespace encryption protects your data at rest, in Oracle RMAN backups, and Oracle Data Pump exports.

Prerequisites and integrations

What identity services are supported today?

With the initial release, Deep Data Security supports local Oracle AI Database end users, Oracle Cloud IAM, and Microsoft Entra ID-based authentication.

Which programming languages/frameworks are supported today?

In the initial release, Oracle Deep Data Security supports JDBC, Python, and ODP.Net client drivers. You can integrate it with commonly used development frameworks such as Spring Boot for Java applications and Django or Flask for Python-based applications.

Do customers have to install an AI Agent or proxy?

No. Oracle Deep Data Security is built into the database kernel. No additional agents or proxies are required. Deep Data Security enforces access at the data layer, limiting bypass opportunities for agentic AI or ad hoc queries.

Is Deep Data Security included in Standard Edition / is Deep Data Security free / is Deep Data Security available on-premises?

Yes, Oracle Deep Data Security is included with Oracle AI Database 26ai Free, Standard, and Enterprise Editions, as well as Oracle AI Database Cloud and Multicloud.

What are the minimum client and driver version requirements for Deep Data Security?

Deep Data Security requires the following minimum driver versions, available with the Oracle AI Database 26ai April Release Update:

- JDBC: Version 23.26.2, supporting JDK 8 through 25

- Python: python-oracledb driver version 4.0 (client driver and plug-in)
- ODP.NET and .NET Core: 23.26.200

Does Oracle Deep Data Security works with Data Pump export?
In the Oracle AI Database 26ai April Release Update, Oracle Data Pump full export does not export Oracle Deep Data Security objects. Customers who require this capability can request a one-off patch through Oracle Support.

For additional information about Oracle AI Database 26ai Deep Data Security, visit: Oracle Deep Data Security.

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