Database Security: A Technical Primer

Seventh Edition

Version 7.26.2 May 2026

ORACLE

Disclaimer

ORACLE

Foreword

Having been in the security space for over 30 years, the front seat view has been exhilarating. Back then, mostly governments and financial institutions were interested in security while everybody else trusted the administrators, users, and computing environment to keep their data secure. It was only when browsers opened up new vistas for commerce over the net in the 90s that companies began to understand the vital need for security. This new perspective led to SSL, network firewalls, and strong cryptography.

Fast forward to the present and, just like before, we find ourselves living in a dramatically different world where every piece of data is online and available 24/7. Now AI is accelerating everything. Generative AI and agentic AI are lowering the cost of attack, from writing convincing phishing messages to searching for misconfigurations, writing exploits, and chaining steps together at machine speed. Some call it "vibe hacking," using AI to rapidly iterate, probe, and persuade until they find the gap you left uncovered.

Over the years, we have seen many different security technologies protecting various layers of the IT stack, from the applications down to the chipsets. Yet attackers have built sophisticated tools, now increasingly AI-enabled, to go after everything we have, whether on mobile devices, laptops, file servers, or databases. For most hackers, the target of choice is not a laptop or a spreadsheet, but a database with hundreds of millions or billions of records. They may try to break in through attacks on the network, applications, operating systems, and databases.

Why are organizations so vulnerable to attacks? Many might say they don't know where their sensitive data is, where they are vulnerable, and what the fixes might be. They might also fear that the fixes may break their applications. Too many organizations still stop at securing the perimeter, not recognizing how easily attackers can bypass it, get to the databases, and quietly walk away with their data. With AI helping adversaries move faster and blend in better, the window between compromise and impact can shrink dramatically. Detection still too often comes late.

To meet this challenge, Oracle has built multiple security technologies focusing on all pillars of security: evaluating the risk posture, preventing attacks from succeeding, and detecting and alerting on malicious behavior. We have enabled customers to deploy security at scale, not just for one database, but for all databases. And because customers cannot spend weeks or months deploying and tuning security controls, we are making it possible to operationalize security at speed. Industry analysts and security professionals recognize that the Oracle AI Database provides the industry's most comprehensive security.

This book, authored by my Database Security Product Management team, explains in simple terms the adversaries of today and how they exploit weaknesses, including how AI changes defender priorities. This book is not meant to be a prescriptive cookbook or a manual but rather a quick study into what every Database or Security Director/VP should know about the security of Oracle databases. You will learn about multiple assessment, preventive, and detective security controls for databases so that you can provide high-level guidance to your teams on how to shrink the attack surface and keep your databases secure at source, at scale, and at speed. I call that Security 360.

Breaches are coming faster than we can imagine, and we must be prepared. Your data is your asset, but unless you protect it well, it could fall into the wrong hands and become a liability. Let's start by securing the source, and do it at scale and at speed.

Vipin Samar
Senior Vice President, Oracle AI Database Security Development

ORACLE

Authors' Notes

Securing data is tough. As security product managers, we hear the same challenge from you constantly: you need to manage real security risk without slowing down databases that run 24/7/365. Whether you are focused on a specific compliance requirement or simply expected to raise your organization's overall security posture, the path forward is not always obvious. This eBook is your security roadmap for the Oracle AI Database, written for security officers, DBAs, application developers, application administrators, system administrators, and security team members.

Rather than listing product highlights, we use a threat-and-solution approach. We show you how attacks unfold, then walk you through the controls that help you stop them. We also know that database security responsibilities are often spread across multiple roles, so we have written this to be useful whether you own the database, the application, or the broader security program. Adversaries do not follow a fixed playbook, and that is exactly why we recommend a defense-in-depth and a Zero Trust mindset.

This edition is significantly updated and expanded, growing from about 60 pages since initial publication to well over 250 pages. We have consolidated and rewritten content throughout to be sharper and more concise, and every chapter reflects the latest Oracle capabilities, current as of May 2026. One of the biggest changes in this edition is the integration of **AI and Agentic AI** throughout the book. This is not a single add-on section tucked away at the back. AI coverage has been woven into every relevant chapter and topic area where it genuinely changes the conversation, whether that is sensitive data discovery, authentication, access controls, or auditing. Because AI's relationship with database security deserves its own focused treatment, we have added a dedicated chapter that addresses AI implications for database security, including how AI functions not just as a tool but as a new attack vector.

Also new to this edition is coverage of **Oracle Deep Data Security**, which supports Agentic AI application development and lets you secure AI-powered workloads with the same rigor you apply to traditional applications. We have also replaced coverage of Oracle Audit Vault and Database Firewall with **Oracle Database Security Central**, which consolidates activity monitoring, security posture management, and controls into a unified platform with new capabilities designed for modern and AI-driven environments.

We have also rethought how the book closes. The final chapter includes a section that walks through each chapter individually, giving you a **fast and convenient refresher of the key ideas**. This is not a recycled collection of the summaries you find at the end of each chapter. It is a fresh synthesis, paired with a "Key takeaways" summary, designed so you can revisit the material quickly without re-reading everything from scratch. After you finish this eBook, you will have a clearer view of how adversaries exploit vulnerabilities, which database security controls protect your databases including controls built for Agentic AI applications, and which risks those controls address. We hope this edition broadens your thinking, gives you actionable guidance, and helps you protect the data that matters most.

Angeline Dhanarani
Bettina Schaeumer
David Knox
Sumanth Kakaraparthi
Ethan Shmargad
Kajal Singh
Manoj Shringarpure
Nazia Zaidi
Pedro Lopes
Peter Wahl
Rich Evans
Roger Wigenstam
Russ Lowenthal
Vasishtha Kumar Nalla

Contents

You Should Read This Book **11**

Yet another book on AI, ransomware and database security 12

The security imperative 12

A framework for defense 13

Regulatory alignment 14

Quantum shadows, rogue AI, and regulatory reckoning 14

Protecting Data **15**

Why database security matters 16

Database security as a growing priority 16

The evolving threat landscape 16

Threat actors and how breaches happen 18

A goal-based approach to data security 21

Choosing between technical and process controls 22

Building a data platform on a converged database 24

Converged security across data types and workloads 25

Summary 26

Database Security Posture Management **27**

The security assessment imperative 28

Understanding the assessment tool landscape 28

The assessment mindset 29

Critical questions attackers seek to answer 29

Assessing configuration security with DBSAT 31

DBSAT report formats and usage 32

Assessing database fleets with Oracle Data Safe 36

Understanding user risk with Data Safe 41

Data Safe deployment and architecture 43

Assessment with Oracle Database Security Central (Security Central) 44

Configuration assessment with EM Database Lifecycle Management 52

Selecting the optimal Database Security assessment tool 55

Minimizing the blast radius with privilege analysis 57

Managing security patches 59

Patching tools and methodologies 62

Hands-on learning with Oracle LiveLabs 63

Summary 64

Sensitive Data Discovery **65**

Introduction: The foundation of data protection 66

Why sensitive data matters 66

The AI imperative: New urgency for discovery 67
Defining sensitive data 68
Discovery methodologies: How to find sensitive data 68
Choosing the right solution 69
Discovering sensitive data using DBSAT 70
Discovering sensitive data using Data Safe 73
Discovering sensitive data using Oracle Database Security Central 76
Discovering sensitive data using Enterprise Manager 78
Hands-on learning with Oracle LiveLabs 82
Summary 83
Authenticating Database Users 84
The importance of strong authentication 85
Unraveling identification, authentication and authorization 85
Database authentication methods 86
Simplify account administration 90
Protect user accounts 90
Hands-on learning with Oracle LiveLabs 92
Summary 93
Controlling Database Access 94
Database privileges 95
Database roles 97
Types of database users 97
Auditing user privileges 98
Centrally managed users 100
Protecting database accounts 101
Hands-on learning with Oracle LiveLabs 103
Summary 104
Enforcing Separation of Duties 105
Why “who can do what” still breaks security 106
What separation of duty means and why it matters 106
Separating the duties 107
Practical tips for a secure database environment 108
Mandatory access controls with Oracle Database Vault 109
Enforcing a trusted path 110
Separation of duties within Database Vault 112
Separation of duties with containerized databases 112
Control SQL database commands with Oracle Database Vault 114
Break-glass scenarios 114
Operationalizing Oracle Database Vault 115

Hands-on learning with Oracle LiveLabs 118
Summary 118
Minimizing Risk from SQL Injection 119
The persistent threat of SQL injection 120
A defense in-depth strategy against SQL injection 121
Database firewall solutions from Oracle 122
Network-based Database Firewall in Security Central 122
Database-resident SQL Firewall in Oracle AI Database 26ai 124
Deciding between Database Firewall and SQL Firewall 125
Hands-on learning with Oracle LiveLabs 127
Summary 128
Data-Driven Authorization 129
The need for data-driven authorization 130
Oracle Database data-driven authorization technologies 131
Virtual Private Database 131
Oracle Label Security 132
Real Application Security 136
Choosing the right technology 138
Hands-on learning with Oracle LiveLabs 138
Summary 139
Masking Sensitive Data 140
Why masking sensitive data matters 141
Understanding data masking 141
Static data masking 142
Data Redaction 143
Data Subsetting 145
Masking formats 145
Masking techniques 147
Implementing static data masking 149
Masking sensitive data using Oracle Data Masking and Subsetting 150
Choosing between Oracle Data Safe and Oracle Data Masking & Subsetting 152
Data Redaction in Oracle Database 152
Implementing Data Subsetting 157
Hands-on learning with Oracle LiveLabs 158
Summary 159
Data Encryption and Key Management 160
Why encryption matters for database security 161
Encrypting data in motion 161
Transport Layer Security (TLS) 161

Encrypting data at rest 162
Transparent Data Encryption 164
Tablespace encryption and column encryption 165
Migrating existing data to encrypted tablespaces 166
Key management 166
Hands-on learning with Oracle LiveLabs 172
Summary 173
Using and Managing Audit Data 174
Why auditing matters 175
Oracle Database unified auditing 175
Audit data best practices 177
What to audit 178
Auditing in the age of AI 182
Managing and using audit data with Oracle Data Safe 182
Auditing with Oracle Database Security Central 188
Hands-on learning with Oracle LiveLabs 195
Summary 196
Securing Agentic AI 197
Security risks in the age of AI 198
Modern risks: vibe coding and shadow AI 198
Vibe coding 199
Why Agentic AI increases the risk 200
Where traditional controls fail 201
Introducing Oracle Deep Data Security 203
Data access control 205
Secure identity propagation 208
Use cases 209
Building a durable AI security strategy 211
Summary 213
Database Security in a Multicloud Environment 214
Understanding multicloud strategies 215
The multicloud security challenge 215
Establishing identity as the foundation 216
Managing Oracle AI Database 26ai in multicloud 216
Leveraging native Oracle services across clouds 217
Enabling secure multicloud service access 217
Implementing multicloud identity management 217
Hands-on learning with Oracle LiveLabs 219
Summary 220

Ransomware and Zero Trust **221**
Ransomware: When data gets taken hostage 222
Understanding the ransomware imperative 222
Defining ransomware and its evolution 223
Phase 1: Immediate high impact defense 224
Neutralizing data theft and extortion through encryption 224
Eliminating data loss through immutable recovery 226
Phase 1 Implementation roadmap 227
Phase 2: The Zero Trust journey 228
Zero Trust: The foundation for long-term resilience 228
Typical Zero Trust implementations 228
Four core Zero Trust projects for databases 229
Zeroing in on Zero Trust 233
Hands-on learning with Oracle LiveLabs 233
Summary 234
Securing the Autonomous Database **235**
Database Security in Autonomous AI Database 236
The value of Autonomous AI Database 236
Security benefits of Autonomous AI Databases 236
Security capabilities of Autonomous AI Databases 237
Shared responsibility 238
Hands-on learning with Oracle LiveLabs 239
Summary 239
Putting It All Together **240**
Oracle Maximum Security Architecture 241
Chapter-by-Chapter journey: Key takeaways 243
Connecting the Controls: How capabilities work together 255
Closing thoughts: Security as an enabler 256
Appendix: Tools – Features, Options, Products, and Packs **257**
Database Security Assessment Tool (DBSAT) 257
Oracle Data Safe 257
Enterprise Manager Database Lifecycle Management 257
Privilege Analysis 257
Native Network Encryption 257
Transport Layer Security 258
Centrally Managed Users 258
Enterprise User Security 258
Traditional Auditing 258
Fine-Grained Auditing 258

Unified Auditing	258
Enterprise Manager Data Discovery	258
Oracle Label Security	259
Oracle Advanced Security	259
Oracle Key Vault	259
SQL Firewall	259
Oracle AI Database Vault	259
Oracle Audit Vault and Database Firewall	259
Oracle Database Security Central	259
Oracle Data Masking and Subsetting	259

Chapter 1

You Should Read This Book

Yet another book on AI, ransomware and database security

You are a busy professional responsible for securing critical data. You do not need another theoretical textbook; you need a field manual.

This book is designed to save time and meaningfully strengthen an organization's security posture. Because most organizations keep their most critical assets in Oracle Database deployments, the focus stays on practical steps that can be applied right away. From strategic planning through configuration, this guide supports steady progress with confidence, helping teams steer clear of expensive security missteps and regulatory dead ends.

Although this book highlights specific features of the Oracle AI Database, the foundations of database security remain universal. The principles described here apply whether the environment runs on-premises, in Oracle Cloud Infrastructure, a hybrid or Multicloud architecture.

The security imperative

The threat environment is not merely expanding; it is accelerating. In 2025, the World Economic Forum estimated the global cybercrime economy at a staggering US$10.5 trillion. Put in practical terms, if cybercrime were a country, it would rank as the world's third largest economy, trailing only the United States and China.

At this scale, private sector measures often struggle to keep pace, and governments step in. The outcome is a global regulatory tsunami. The United States alone enforces more than 100 privacy and data security laws. In the European Union, the General Data Protection Regulation (GDPR) is now reinforced by the Digital Operational Resilience Act (DORA) and the updated NIS2 Directive. India has introduced the Digital Personal Data Protection Act (DPDPA), and more than 130 other nations have enacted similar privacy legislation.

The stakes are financial, not only reputational. GDPR violations can trigger fines of up to 4% of global annual turnover or €20 million. Since 2021, fines have exceeded €4.5 billion. For any business operating internationally, this can quickly become a minefield of concurrent and overlapping regulatory obligations.

It's not just regulations

Data risk is a multi-front war. Organizations must contend with stricter laws, rising privacy expectations, and a relentless barrage of attacks:

- Ransomware: This remains a primary concern, accounting for nearly 17% of all cyberattacks and costing tens of billions annually.
- Geopolitical Instability: Nation-state actors actively target private data for economic advantage or to weaken critical infrastructure.
- AI: As Agentic AI systems expand across enterprises, they bring major productivity gains while extending the security perimeter in ways many organizations are still working to map, measure, and manage.

These trends show no signs of slowing. The time to protect data is now, before the breach, not after.

The hidden opportunity: agility

Regulatory pressure and data theft risk often launch database security programs, but a third driver is equally important: **agility**. Customers increasingly factor security posture into purchasing decisions. A consistent, successful security program reduces sales friction, speeds due diligence, and strengthens win rates.

The same applies outside commercial settings. For organizations dependent on grants, public funding, or private investment, a strong security track record builds trust and shortens reviews. Stakeholders are more likely to support organizations that proactively avoid breaches and negative headlines.

Robust security is not only a shield; it is a strategic enabler that helps organizations move faster and seize new opportunities.

A framework for defense

Thousands of regulatory requirements can feel like a wall of obligations. But trying to enumerate every rule rarely helps, and quickly becomes unmanageable. `Disclaimer: This is not legal advice; always consult your legal counsel for compliance guidance.`

A more practical path is to look at the regulatory landscape as a framework of controls. Most regulations are trying to solve the same underlying problems. No matter the acronym, whether GDPR, PCI DSS, HIPAA, they typically boil down to three fundamental actions that must take place on your data:

1. **Encrypt** your data.
2. **Control access** to your data.
3. **Audit access** to your data.

Start with these outcomes. Once these pillars are in place, it becomes much easier to layer on the reporting and documentation that each regulation asks for.

1. Encrypt your data

Encryption is often your last line of defense for data at rest and backups. It transforms your data using cryptographic algorithms, rendering it unreadable without the correct decryption key. Even if an attacker bypasses your firewalls and steals your storage media, encrypted data remains useless to them. Database files and database backups are prime targets.

Effective encryption is intrinsically linked to key management. Keys need to be generated, stored, rotated, and retired securely. Without strong key management, encryption is like installing a sturdy lock and then taping the combination to the door.

2. Control access to your data

Access control acts as the gatekeeper. It decides who can get into the database and what they are able to do once they are there. This broad area usually breaks into three layers:

- **Authentication**: Verifying the identity of the user or service (Checking the ID card).
- **Authorization**: Defining which objects a verified user is permitted to view or touch (Assigning the keycard).
- **Fine-grained Access**: Pinpointing exactly which rows or columns in a table are visible (Redacting specific lines in a file).

When least-privilege is enforced, users receive only what they need and nothing more, which dramatically limits the blast radius if an account is compromised.

3. Audit access to your data

If access control is the gatekeeper, auditing is the security camera. It records what happens across systems so actions can be traced, issues can be investigated, and compliance expectations can be met.

A solid audit trail answers the classic forensic questions:

- Who performed the action? (Database account and OS user)

- What did they do? (Insert, update, delete, or execute)
- When did it happen? (Timestamp)
- Where did they come from? (Source IP or hostname)
- How did they connect? (Client program)

Auditing will not stop the incident, but it gives the evidence needed to reconstruct what happened and show regulators that the environment is being actively watched.

Regulatory alignment

Beyond these three foundational controls, encryption, access control, and auditing, some regulations may impose additional requirements. However, most expect these basics to be implemented first. Table 1-1 provides examples of regulations that mandate these and other types of controls.

Regulation	Country	Required control	Remarks	Applies to
PCI DSS	Global	Redaction - A specialized form of access control.	Only part of a credit card number should be visible.	Anyone processing credit cards
DORA	European Union	Zero Data Loss Recovery	Implicit goal is recovery from ransomware attacks.	Financial institutions in the EU
Companies Act (Rule 11)	India	Before/After value tracking - A specialized form of auditing	Track changes to entries in books of account.	Companies using software for accounting.
Sarbanes-Oxley (SOX)	United States	Separation of Duties - A specialized form of access control.	Ensure financial records are not altered by unauthorized personnel.	Publicly traded companies.

Table 1-1: Summary of regulatory-influenced controls

Quantum shadows, rogue AI, and regulatory reckoning

The security landscape of 2026 has shifted from passive defense to active warfare. Ransomware has moved beyond simple encryption. Attackers now use “harvest now, decrypt later” strategies, collecting data today and waiting for quantum computing to break traditional protections.

At the same time, the rise of Agentic AI introduces a paradox. These autonomous agents can unlock remarkable productivity, yet they can also expand the attack surface exponentially. An agent built to execute complex business logic can be manipulated through prompt injection and turned into a high-speed insider threat, exfiltrating sensitive data before a human operator can react.

The regulatory environment is tightening its grip to match these threats. 2026 marks the entry of strict new data privacy laws, joining a global mesh of “Universal Opt-Out” mandates that force businesses to honor consumer privacy preferences automatically. Compliance is no longer a quarterly checklist; it demands real-time action. The cost of failure has escalated from uncomfortable fines to existential threats, with penalties under frameworks like GDPR and the new US state statutes capable of erasing annual profit margins overnight. In this fragmented regulatory map, your database is either your strongest asset or your greatest liability.

Defending against machine speed attacks requires more than vigilance. It calls for embedded intelligence such as the Oracle AI Database 26ai, with quantum-resistant cryptography and ML-driven SQL Firewalls intended to help distinguish between a legitimate agent and a compromised one.

That is why this book matters. It avoids abstract theory, and focuses on practical outcomes. Turn the page, and let’s get to work!

Chapter 2

Protecting Data

Why database security matters

Databases store an organization's most sensitive assets: trade secrets, personal records, financial transactions, and customer intelligence. When breaches or ransomware hit, the fallout is severe. For security administrators, database owners, and data custodians, database security must stay a daily priority.

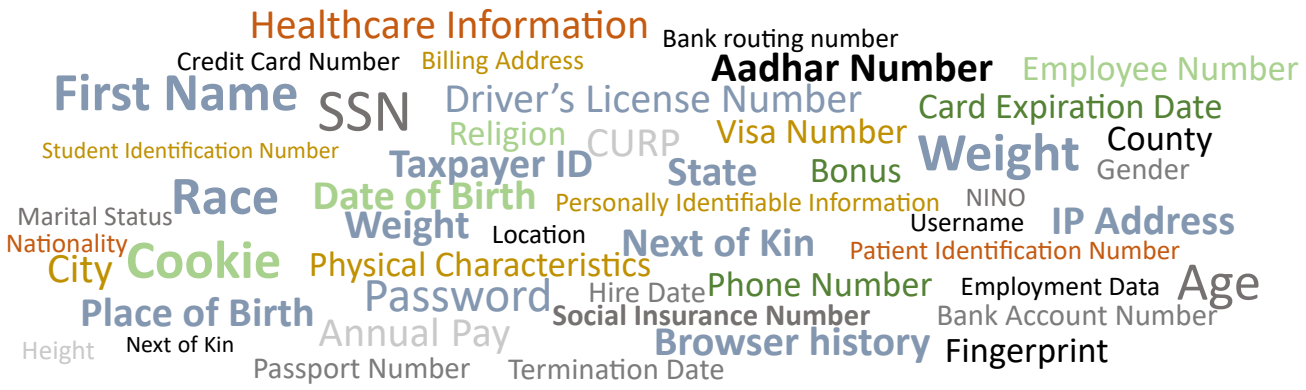


Figure 2-1: Many types of personal data demand comprehensive database security

This book guides readers through a defense-in-depth security strategy for Oracle Database, demonstrating how various controls work together and what protection they deliver.

Database security as a growing priority

The database administration landscape has transformed dramatically in recent years. At a recent Oracle AI World event, database administrators from a large financial institution revealed that security now consumes over half their workday. A decade ago, security barely registered as a concern. Today, regulations such as GDPR, CCPA, and India's DPDPA, combined with the ransomware epidemic, emerging threats from Agentic AI, and daily breach headlines, have elevated security to a top organizational priority.

Many database professionals now ask, "What matters most for keeping our data safe?" This question reflects widespread recognition that database security cannot be treated as an afterthought.

There is no magic formula for perfect security. Organizations face tireless, creative adversaries who continuously seek opportunities to compromise data. The objective is not simply checking compliance boxes but rather identifying vulnerabilities and staying ahead of evolving threats. Understanding exposure points is essential because attackers consistently target the weakest links in an organization's defenses.

Threats take many forms. Less sophisticated attackers exploit outdated systems with readily available tools. Advanced persistent threat actors move laterally through networks, probing for vulnerabilities and extracting data silently over extended periods. Sometimes the threat comes from within, such as a contractor or employee turned adversary. These breaches can remain undetected for months or years, making constant vigilance essential for effective defense.

The evolving threat landscape

Three interconnected forces drive the increasing urgency around database security: regulatory requirements, sophisticated threat actors, and emerging technology risks. Understanding these factors helps security teams prioritize their efforts and allocate resources effectively.

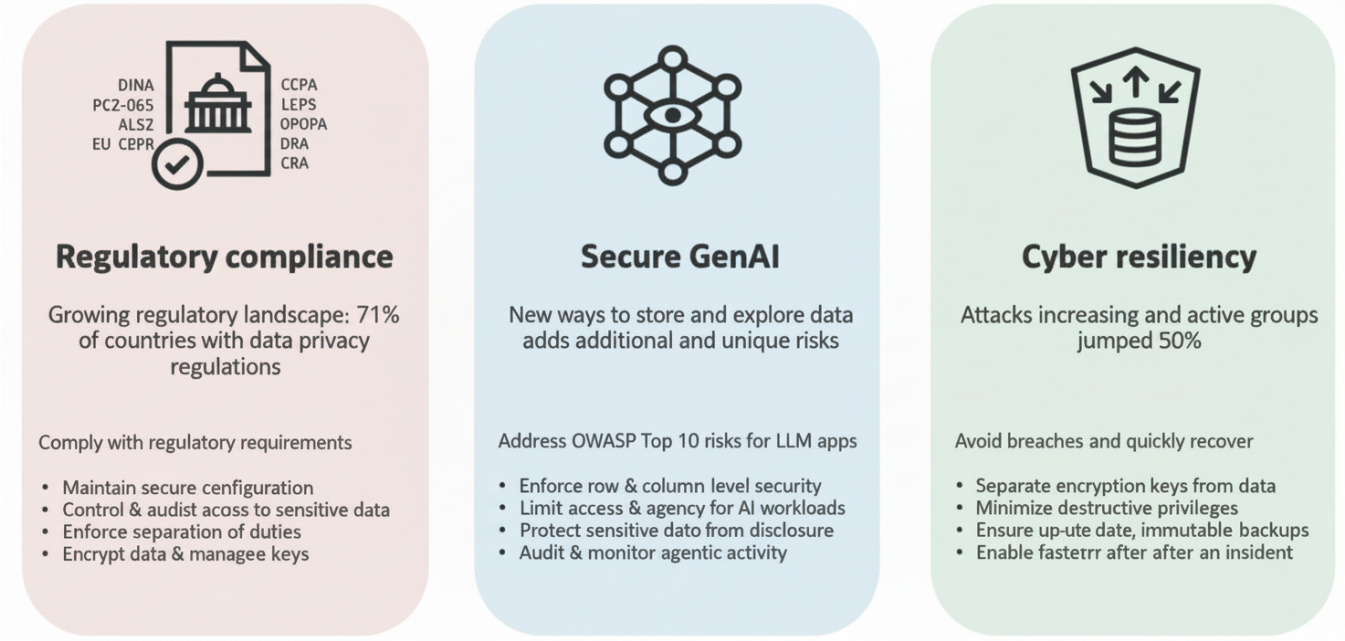


Figure 2-2: Database security drivers for an evolving compliance and threat landscape

Regulatory compliance continues to expand globally. Data protection regulations now span multiple jurisdictions, each with distinct requirements for consent, breach notification, data residency, and individual rights. Organizations operating internationally must navigate complex overlapping mandates while maintaining consistent security practices across their database environments.

Threat actor sophistication has increased substantially. Attackers range from opportunistic attackers using automated tools to nation-state actors conducting targeted campaigns. Ransomware groups now combine data encryption with exfiltration and extortion, threatening to publish stolen data unless organizations pay. These groups often target backup systems and disaster recovery infrastructure to maximize leverage. In response, organizations are increasingly prioritizing cyber resilience, building the capacity not only to prevent attacks but to absorb, adapt to, and rapidly recover from them when prevention fails.

Emerging AI threats represent a new challenge for database security. Agentic AI systems that can autonomously interact with databases introduce novel attack vectors. Adversaries may leverage large language models to generate sophisticated SQL injection payloads, automate reconnaissance of database structures, or identify sensitive data patterns at scale. Conversely, AI-powered defensive tools offer opportunities to detect anomalous behavior and respond to threats more rapidly than traditional rule-based systems.

Database security in the age of AI

Enterprise technology is changing quickly, and AI and Agentic AI represent a major turning point from generating content to taking action autonomously. In this new era of database security, it is important to keep in mind that AI agents can plan work and interact directly with core systems, widening the security perimeter in ways older models may not fully reflect. The practical takeaway is clear: the goal is no longer just protecting applications but governing AI-mediated actions with the same discipline used for human users.

Figure 2-3 makes the dependency chain for effective AI clear. AI runs on Data (capital “D”) as its fuel, and the more fuel, or Data, it can draw on, the greater the potential benefit. That dynamic puts a spotlight on the security and reliability of the data feeding AI. Reliability is not just about availability; it is also about whether the data can be trusted and whether it accurately reflects what it is meant to represent. Security matters just as much, particularly because

sensitive information can be disclosed inadvertently. Even more troubling is the risk that data could be improperly changed or deleted.

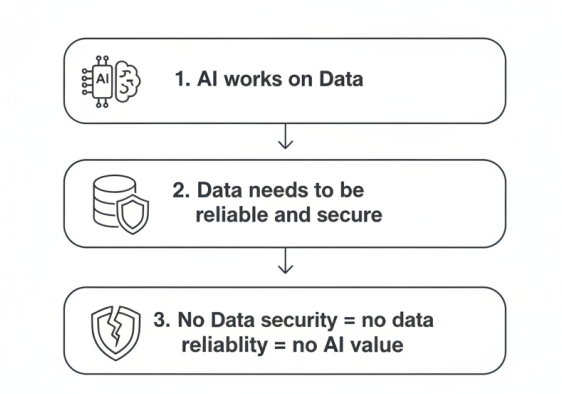


Figure 2-3: Effective AI depends on Database Security

AI can amplify database risk simply by making application development faster. One example is 'vibe coding,' where developers lean on the speed of AI-generated code without taking the time to fully understand what it is doing. The productivity gains are real, but the tradeoff is often black box logic that can quietly introduce SQL injection vulnerabilities, expose sensitive API keys, or create unintended data exfiltration pathways. Seeing these risks clearly enables organizations to move from default trust to deliberate verification, validating every AI-generated query against established security standards before relying on it.

Then there is Shadow AI, which adds a different kind of governance challenge. Well-meaning employees may connect unauthorized AI tools directly to corporate databases to move faster. Unlike traditional shadow IT, autonomous agents can make their own calls about sensitive information, which can lead to proprietary data leaking across organizational boundaries without an audit trail. This section underscores why broad bans often fail in practice and why securing the data at its source matters most, ensuring protection remains unmovable no matter which tool is used.

The most durable path forward is to treat database-native enforcement as the primary control plane. Layered defenses such as Transparent Data Encryption (TDE), fine-grained access controls like Virtual Private Database (VPD), and SQL Firewall that block unauthorized statements help contain the blast radius of runaway agents or prompt injection attacks. With these controls in place, organizations can pursue AI-driven advantages with confidence, keeping innovation resilient, compliant, and under control.

Now that we've named these AI-driven risk amplifiers, here's how database-native controls contain them.

Threat actors and how breaches happen

In the past, perimeter defenses provided a sense of security. Today's threat landscape evolves too rapidly for any single barrier to suffice. Set-and-forget security is no longer viable. Organizations need layered controls across the

application and data stack. If one layer is breached, others must remain intact. This approach, known as defense-in-depth, remains one of the strongest security strategies available.

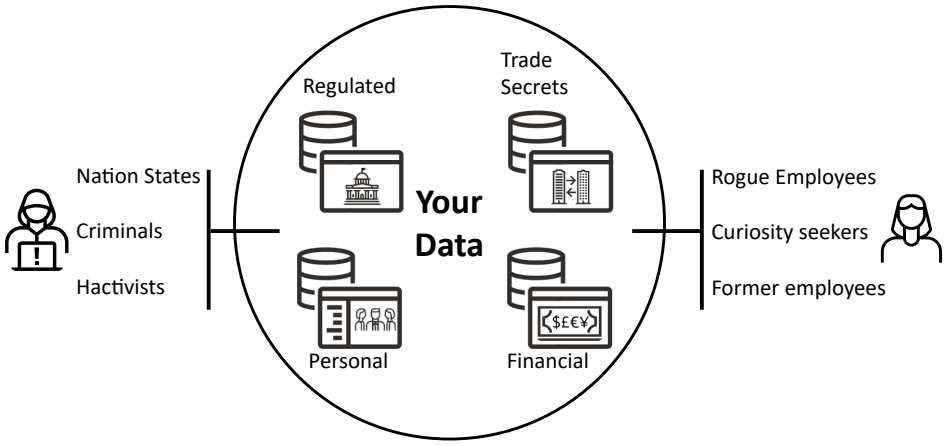


Figure 2-4: Threat actors have many motivations compounding the challenge to securing data

Attackers compromise database security through multiple pathways. Security professionals are likely familiar with the OWASP Top Ten, the widely refernced catalog of web application vulnerabilities that evolves continuously as threats change. Similarly, the database security community tracks a "Dirty Dozen" list of the most common methods attackers use to compromise database security.

Common attack methods and vulnerabilities

Understanding how attackers operate enables security teams to focus on closing real gaps and improving organizational resilience. The following tactics represent some of the most prevalent methods from the attacker's playbook:

- Exploiting unpatched systems or misconfigured databases to bypass access controls
- Escalating runtime privileges by exploiting vulnerable applications
- Searching for sensitive data in unprotected databases, applications, and systems
- Stealing the credentials of privileged administrators or application users
- Accessing accounts through password guessing or exploiting careless credential management
- Exploiting application weaknesses with techniques such as SQL injection
- Exploiting unprotected systems as a bridge to launch attacks against more sensitive targets
- Creating rogue user accounts on systems as a base for reconnaissance and privilege escalation
- Targeting copies of live production data used in development and test environments where protections are typically weaker
- Accessing unencrypted database system files on disk or in backup files
- Locking data with encryption and demanding ransom payments for decryption keys

These attack methods correspond to a comprehensive list of database security risks known as the "Dirty Dozen." Table 2-2 below provides the complete inventory.

THE DIRTY DOZEN

1. Insecure configuration and configuration drift
2. Unpatched and out-of-date systems
3. Lack of a consistently enforced security policy
4. Lack of visibility into sensitive data placement and quantity
5. Overprivileged database users and administrators
6. Weak authentication and shared accounts
7. SQL injection vulnerabilities and insecure application design
8. Trusting vulnerable networks
9. Insufficient or inefficient monitoring and auditing
10. Sensitive data proliferation to non-production databases
11. Unprotected servers and database backups
12. Insecure encryption keys and secrets

Table 2-2: The Oracle Database Security Dirty Dozen represents the most common database security risks

Attackers exploit risks at every touchpoint with the database. They most frequently target accounts with significant database access, whether end users, application service accounts, or administrators. Stolen login credentials remain a common entry point. SQL injection remains a popular attack vector because penetrating an application is often easier than gaining direct access to the database or its server. When attackers gain access to network segments carrying database traffic, they may intercept unencrypted communications to extract sensitive data.

Once an attacker establishes a session with the database or accesses the server directly, they may exploit lingering unpatched vulnerabilities or attack the database from within the server environment. A now-prevalent technique is data scraping: extracting and exporting raw database files without establishing a formal session. This method is common in most ransomware data breaches.

Database copies created for testing or development are equally at risk. In targeted campaigns where attackers pursue entry into a specific known database, adversaries frequently focus on these non-production copies because they are usually less secure and less monitored than production counterparts. Additionally, these environments are often accessed by a wider user community, including developers, testers, and third-party contractors, further expanding the attack surface.

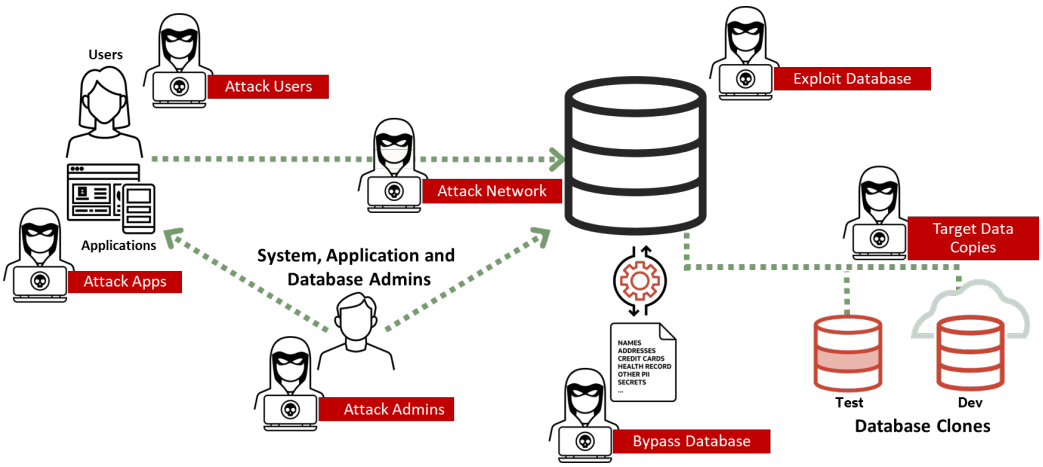


Figure 2-5: Hackers can attack the database from many directions

A goal-based approach to data security

A well-structured data security posture helps mitigate the risks outlined in the Dirty Dozen. Successful approaches incorporate multiple layers of security controls to provide defense-in-depth protection against threats. Database security objectives can be organized into four broad categories:

1. Assess security posture
2. Control access to data
3. Monitor user activity
4. Protect data against theft

To mitigate database security risks effectively, organizations should set clear security goals and align them to actionable tasks. A goal describes the intended security outcome, such as ensuring secure user authentication, while a task is the concrete step taken to achieve that goal, such as implementing multifactor authentication. Table 2-3 provides examples of high impact security tasks, prioritized by how effectively they reduce risk.

Category	Goal	Task	Threat addressed
Assess security posture	Ensure systems do not deviate from their approved security baseline	Monitor databases for configuration changes that introduce risk	Insecure configuration
Monitor user activities	Detect and deter inappropriate activity by privileged users	Monitor and audit privileged user activity	Insufficient or inefficient monitoring and auditing
Protect data against theft	Remove risk from test and development environments	Scramble sensitive data in test and development environments	Sensitive data proliferation to non-production databases
Control access to data	Strengthen user authentication to prevent compromise of accounts	Implement multifactor authentication for the database	Weak authentication and shared accounts

Table 2-3: Examples of high-level security goals and tasks

If teams are spending time on tasks that do not clearly map to a specific security goal, it usually signals one of four scenarios:

1. **External regulatory mandate**: The task exists to satisfy an externally mandated regulatory requirement. Whether it neatly aligns to internal goals is secondary because compliance is nonnegotiable. **Action**: Complete it as efficiently as possible, then get back to value-added work.
2. **Internal regulatory mandate**: The task meets an internally mandated regulatory requirement. **Action**: Ensure everyone has a shared understanding of both the goals and the requirement. If the task genuinely does not contribute, explore whether the organization is willing to adjust the requirement and redirect time to higher value work.
3. **Goal drift**: The goals were once sound, but they no longer match today's reality. In security, that is common as technology advances, tactics shift, and new threats appear every day. **Action**: Revisit the goals and reprioritize based on current risk.
4. **Misdirected effort**: Time and resources have landed on a task that provides little or no benefit. **Action**: Depending on the context, either continue if it meaningfully supports another team's goals, or pause and pivot to work that directly reduces risk.

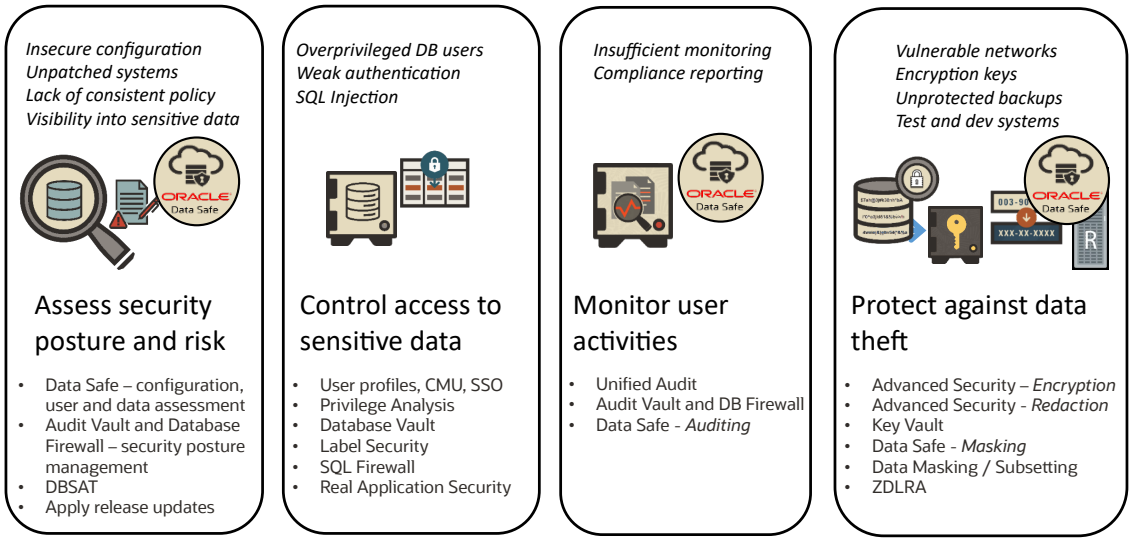


Figure 2-6: A goal-based approach to combatting the Dirty Dozen

Once goals are clear and the supporting tasks are mapped out, the last step is choosing and implementing the most suitable control, whether it is technology driven, process driven, or a blend of both.

Choosing between technical and process controls

For most Oracle Database tasks, there is rarely a single right way to execute. Take monitoring databases for configuration changes that introduce risk, the first example in Table 2-3. Organizations typically have several control options:

- **Process control (policy-based)**: Set the expectation in writing. Publish a policy that mandates a specific database configuration and prohibits changes, then require each database administrator to acknowledge and agree to comply. This approach succeeds or fails based on consistent adherence to the process.
- **Process control augmented by technology (scripted check)**: Add an objective checkpoint. Write a script that checks security critical database parameters, then run it across all databases on a regular cadence to

verify settings. This is more effective than policy alone because it measures what is actually in place. In this model, the script is the technical control, and the process control is the policy that requires running it and reviewing the results.

- **Process control augmented by technology (manual tool)**: Use an established assessment tool instead of building from scratch. Manually run the Oracle Database Security Assessment Tool (DBSAT) or a third-party tool like the Center for Internet Security (CIS) benchmark, then review the reports on a schedule. This pairs a technical control, DBSAT, with a process control, the policy to run the tool periodically and act on findings. Vendor supplied tools are often faster to adopt than custom scripts because the vendor has already done the work of identifying relevant parameters and settings.
- **Pure technical control (automated tool)**: Automate the work end to end. Deploy a tool that continuously or periodically scans databases for security issues, schedule checks daily, weekly, or monthly, and enable email alerts when the security posture drifts. Examples include Oracle Data Safe, Oracle Database Security Central, or other Database Security Posture Management (DBSPM) tool. This option minimizes dependence on manual follow through and is typically stronger than manual approaches because scanning and change detection happen automatically.

In these examples, the control types can be summarized in Table 2-4 as follows:

Approach	Control Type
Publish a policy	Process control
Write a script	Process control supplemented with technology (custom script)
Download and manually run a tool	Process control supplemented with technology (vendor's tool)
Purchase a tool that automates the process	Technical control

Table 2-4: Summary mapping of approaches to technical and process control types

The middle two options sit in the sweet spot between process and technology. They use a technical component, a script or a tool, but they still rely on people to run it and interpret the results.

It is also important to be candid about process-only controls: they are often ineffective. Auditors evaluate process-based security controls by sampling a statistically relevant subset of databases. In smaller environments, such as those with 100 databases, auditors may verify every database. In larger deployments with thousands of databases, auditors typically select a representative random sample, often around one or two percent, to gauge compliance.

With a process control supplemented with technology, auditors typically do three things. They validate the tool's effectiveness, whether it is widely used in the marketplace or developed internally. They review the process for running the tool. Then they sample a statistically relevant set of databases to confirm the tool was run and the results were reviewed.

With a purely technical control, auditors still validate the tool's effectiveness, but they also confirm it covers all databases. This path is usually faster than evaluating a purely process-driven control, and it often reduces audit costs because there is less manual evidence gathering to review.

Practitioners who regularly work with external auditors have likely seen this firsthand. A process control can pass an audit, but it usually takes auditors much longer to gain confidence in its effectiveness.

Note: The ISACA Certified Information Systems Auditor (CISA) course of study is highly recommended for professionals who work with security assessments and audits. The CISA certification offers valuable insight into running an effective assessment program.

Finally, it is worth remembering that technology is not automatically the best answer. In a scenario with a single Oracle Database instance, the effort to implement and maintain a purchased solution would likely outweigh the effort of downloading DBSAT and running it weekly. In very small environments, a technology supported process control can be the more practical choice over a pure technical control.

Building a data platform on a converged database

The remaining chapters focus on the security capabilities built into an Oracle AI Database. It is important to recognize that the Oracle AI Database often sits at the heart of many IT data platform architectures. Because it is a converged database that supports multiple data models, one of its most valuable traits is that the security covered in the rest of this book stays consistent across these multimodal data types.

Solving data inefficiencies with data convergence

The Oracle AI Database differentiates itself by bringing multiple data paradigms, including relational, JSON, graph, spatial, and vector, into a single high-performance engine. This consolidation helps eliminate the architectural complexity and fragmentation that typically come with operating several specialized databases. With this unified approach, see Figure 2-, enterprises can replace disconnected data silos with a cohesive platform that supports innovations such as JSON Relational Duality, where data is stored once and accessed through multiple models without the latency or risk introduced by complex ETL pipelines.

Importantly, this architecture extends beyond on-premises and OCI; it also runs natively on Amazon Web Services, Microsoft Azure, and Google Cloud. As a result, organizations can maintain a consistent operational and security posture across environments, which can simplify compliance with stringent data residency and sovereignty requirements.

By collocating accurate, multimodal data with applications on any hyperscaler, Oracle offers a simplified, secure 'many-as-one' approach that significantly lowers administrative overhead while continuing to deliver the high availability, security, and low latency required for mission-critical AI and distributed cloud workloads.

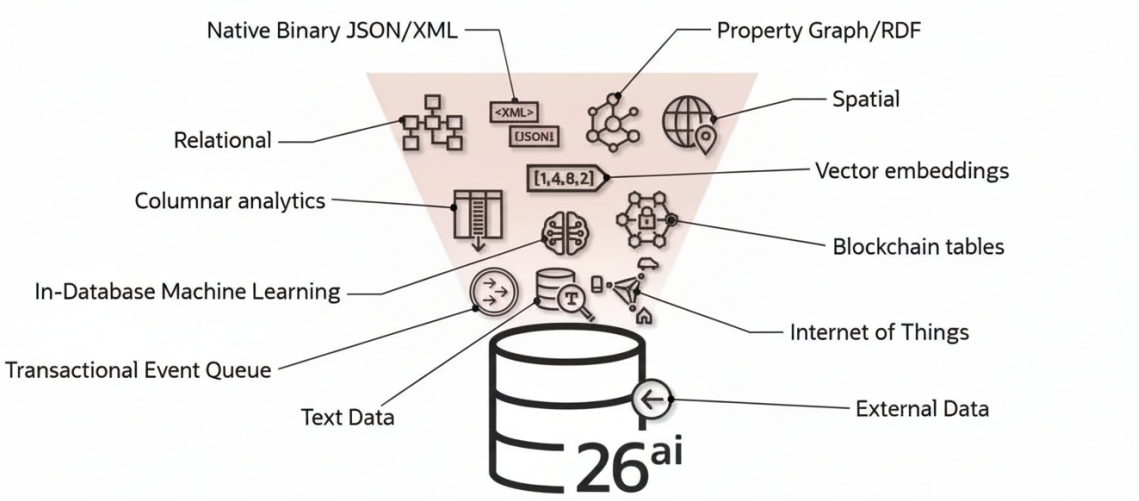


Figure 2-7: Oracle Converged Databases manage and secure "many-as-one"

Converged security across data types and workloads

Oracle's converged database strategy enhances data security by enforcing consistent policies across all data types and workloads. When organizations rely on a collection of specialized database engines for workloads like graph, spatial, JSON, or time-series data, they inherit a fragmented security landscape that is difficult to manage and even harder to defend. Each engine reaches maturity at a different pace: some offer robust encryption, fine-grained access controls, and comprehensive audit logging, while others lag behind and leave exploitable gaps. Attackers don't target your strongest system; they target your weakest one.

Fragmentation increases operational burden and expands the attack surface

Fragmentation multiplies the workload for security teams. Instead of defining a policy once and applying it universally, administrators must configure, monitor, and patch each engine independently, translating the same requirements into the syntax, tooling, and conventions of each platform. That translation introduces inconsistency, and inconsistency introduces risk. A role-based access control policy that works exactly as intended in one engine may behave subtly differently in another, and those subtle differences are precisely where breaches happen.

Specialized engines also tend to require data movement. Data is often extracted from a source system and loaded into the engine best suited to process it. Every time data moves, it crosses a boundary that must be secured, monitored, and audited. Pipelines, staging environments, and integration layers expand the attack surface and create new points of failure.

Bolt-on security versus security built into the core

Many specialized engines share a fundamental architectural weakness: security that is bolted on rather than built in. These engines are often optimized first for performance or narrow functionality, with security added later as a layer on top. The result can look sufficient on paper but lack the depth needed to withstand sophisticated threats. Encryption may not extend through the full stack, access controls may not apply uniformly across operations, and audit trails may have blind spots. Across a fleet of specialized engines, each with its own bolt-on security layer of varying quality, the cumulative exposure is significant.

Oracle’s integrated approach and its benefits

Oracle takes a fundamentally different approach. Security in Oracle AI Databases is architected into the core of the engine. Capabilities like Transparent Data Encryption, Database Vault, Label Security, Unified Auditing, and Data Redaction have been developed, hardened, made to scale, and refined over decades as first-class features. Because they are deeply integrated, they apply consistently across every workload, every data type, and every operation the database performs. There are no gaps at the seams and no inconsistencies between how security behaves for a relational query, a graph traversal, or a JSON document operation.

This integration delivers concrete benefits. Security controls cannot be easily bypassed by approaching the data through a different interface or workload type. Audit logs are complete and trustworthy because the instrumentation sits inside the engine. Compliance is easier to demonstrate because evidence of control effectiveness is consistent and centrally accessible. Security teams also spend less time compensating for immature or shallow implementations across disparate systems and more time on work that reduces risk.

Taken together, Oracle's converged database approach minimizes data movement between systems, eliminates the inconsistencies of managing security across fragmented specialized engines, and replaces bolt-on security of varying maturity with a single, deeply integrated, battle-tested security architecture. This simplifies administration while delivering a materially stronger and more defensible security posture.

Summary

Protecting data isn't just another box to tick, it's the bedrock of modern database operations. Databases concentrate an organization's most sensitive assets in one place: trade secrets, customer intelligence, and personal and financial records. And the stakes have only gotten higher. Security has shifted from a background concern to a daily priority for many database teams, pushed forward by expanding regulations like GDPR, CCPA, and India's DPDPA, plus ransomware and the steady drumbeat of breach headlines. The real aim is to stay ahead of tireless adversaries by learning where exposure likes to hide and by staying vigilant, not by sprinting from one compliance checklist to the next.

To do that well, it helps to understand how breaches actually unfold. Attackers often follow familiar paths: stealing credentials, slipping in through SQL injection, intercepting unencrypted traffic, exploiting unpatched vulnerabilities, or using ransomware playbooks that blend encryption with exfiltration and extortion. One especially dangerous modern pattern to keep on your radar is data scraping, where attackers extract raw database files without ever establishing a formal session. This technique is described as common in many ransomware breaches. Also watch the "quiet liability" that creeps in through convenience: less monitored copies of production data in development and test environments. These tend to have weaker protections, and attackers know it.

When you move from awareness to action, anchor your efforts to outcomes and choose controls that reliably reduce risk. A practical way to plan is to use a goal-based structure across four categories, then apply a clear decision mindset for selecting controls. Weigh options like policy-only approaches, scripts, manually run assessment tools such as DBSAT or CIS benchmarks, and fully automated tools that continuously scan and alert when configurations drift. Keep audit realities in view, too. Auditors sample environments, so evidence and coverage matter. Purely technical controls can shrink audit effort when they demonstrably cover all databases. At the same time, if your environment is small, you may get better results with lightweight, technology-supported process controls.

With defense-in-depth established and the Dirty Dozen risks made concrete, the next practical question is unavoidable: how do you keep hundreds of security levers aligned across real, messy fleets where drift is inevitable and attackers scan at machine speed? That's where Database Security Posture Management comes into focus, treating security assessment as systematic reconnaissance. It's about finding misconfigurations before they become headlines, proving due diligence to auditors and regulators, and building evidence that security is improving over time.

Oracle AI Database commonly serves as the core of enterprise data platforms. As a converged database, it supports relational, JSON, graph, spatial, and vector data models in a single, high-performance engine. This consolidation reduces the complexity and inefficiency of managing specialized databases, while minimizing data silos and ETL-heavy pipelines. JSON Relational Duality supports storing data once and accessing it through multiple models. Running across on-premises, OCI, and major hyperscalers (AWS, Azure, Google Cloud) also helps organizations keep operations and security consistent across environments, supporting requirements such as data residency and sovereignty.

From the security viewpoint, specialized databases create fragmentation which leads to inconsistent security, higher operational burden, and a larger attack surface, especially when data movement introduces additional boundaries and failure points. Making things worse, many specialized databases rely on "bolt-on" security that can be uneven or incomplete. Oracle instead embeds security in the core engine, applying long-established capabilities (e.g., Transparent Data Encryption, Database Vault, Label Security, Unified Auditing, and Data Redaction) consistently across all data types and workloads. This approach reduces bypass risk, improves audit completeness, simplifies compliance evidence, strengthens security, and lowers administrative overhead.

Turn the page to see how posture management transforms database security from a hopeful checklist into a repeatable, scalable discipline, with tools and workflows designed to keep every gate latched, every lock tested, and every weak setting surfaced and remediated before attackers can exploit it.

Chapter 3

Database Security Posture Management

The security assessment imperative

Database security depends on hundreds of individual configuration decisions, and each one affects your overall security posture. A single misconfigured parameter, a weak password policy, an overprivileged account, or disabled auditing can create an exploitable weakness. These vulnerabilities hide across layers of configuration complexity, so teams that rely on periodic manual checks are already behind. A systematic security assessment lets you find and fix gaps before attackers do.

Attackers do not wait for an invitation. They use automated tools to probe thousands of systems simultaneously, searching relentlessly for that one misconfiguration that opens the door. They are patient, methodical, and persistent, and the cost of giving them what they are looking for is steep.

Configuration drift has consequences that extend far beyond a single breach. Data loss, prolonged downtime, brand damage, regulatory fines, and executive accountability can all trace back to preventable security gaps. When your organization stores personal data such as names, addresses, birth dates, and account details, regulators expect documented controls and ongoing verification. Laws and frameworks including GDPR, PCI DSS, Sarbanes-Oxley, and breach-notification statutes require regular security assessments for any system that processes personal data.

A strong assessment practice accomplishes three things:

1. **Identify misconfigurations before attackers exploit them.** Proactive detection stops breaches before they start instead of forcing teams to respond after the damage is done.
2. **Demonstrate due diligence to regulators and auditors.** Documented assessments show ongoing security investment and provide verifiable proof of compliance.
3. **Build defensible evidence of improvement over time.** Trend analysis shows measurable risk reduction and supports a continuous improvement cycle.

A standards-based approach brings together three essential elements: relevant regulations and security frameworks, Oracle Database hardening best practices, and your organization's specific policies and risk priorities. Together, they create a repeatable, defensible process to identify misconfigurations, prioritize critical issues, and demonstrate measurable progress.

This chapter explains how Oracle Database security solutions help you rapidly assess your security posture, categorize findings by risk and compliance relevance, and define clear remediation paths. You will see which configurations to address first, why they matter, and how to fix them so you can harden your databases faster, streamline audits, and sustain strong security without slowing your business.

Understanding the assessment tool landscape

Organizations assessing Oracle Database security have access to multiple tools, each with distinct capabilities, deployment models, and use cases. Some tools ship with Oracle Database, while others are available as add-on products or cloud services. Most Oracle Database customers already have access to one or more assessment options through existing licenses or enterprise agreements.

The key is matching tool capabilities to organizational goals, constraints, and scale. Four primary tools address database security assessment:

1. **Database Security Assessment Tool (DBSAT)**: A command-line utility provided at no cost to all Oracle Database customers with active support agreements
2. **Oracle Data Safe**: An Oracle Cloud Infrastructure service included with all Oracle Cloud database services, also available for on-premises databases, third-party cloud deployments, and Oracle Database@Multicloud

3. **Oracle Database Security Central (Security Central)**: A database security posture management platform available for on-premises or Oracle Cloud deployment
4. **Oracle Database Lifecycle Management Pack (DBLM)**: A management pack for Oracle Enterprise Manager

This chapter outlines each tool's function, strengths, and trade-offs to help organizations select the optimal fit for their environment, budget, timeline, and operational model.

The assessment mindset

Before looking at assessment tools, it is worth stepping into an attacker's mindset, because that perspective makes it easier to prioritize the right security controls. Attackers treat databases like seasoned burglars surveying a neighborhood. They do not kick in doors randomly. They watch patrol routines, try the doorknobs, notice which houses have broken security cameras, and plan the fastest way out. Today, many of them go further, using AI-powered scanners to sweep thousands of databases at the same time, all in pursuit of that one unlocked door.

The attacker's reconnaissance toolkit

Attackers utilize automated tools and techniques to systematically identify vulnerabilities:

- **Database discovery**: Network scanning identifies database instances, versions, and network exposure
- **Port and service scanning**: Open listeners and exposed management interfaces reveal attack surfaces
- **Vulnerability identification**: Automated tools match database configurations against known CVE vulnerabilities
- **Application and SQL injection testing**: Web application proxies probe for input validation weaknesses
- **Brute-force authentication attacks**: Password spraying and dictionary attacks test for weak credentials

Following this reconnaissance phase, attackers focus on the weakest links and formulate efficient paths to data. They assess an organization's security posture to find the path of least resistance. Understanding these tactics helps security teams and DBAs prioritize hardening steps that make answering critical attacker questions substantially harder.

Critical questions attackers seek to answer

When probing database defenses, attackers focus on specific questions that reveal exploitable weaknesses:

- **What database version is running?** Outdated versions with known vulnerabilities become primary targets
- **Are default or weak credentials in use?** Unchanged default passwords and weak password policies provide immediate access
- **Which privileged users exist?** High-privilege accounts enable privilege escalation and lateral movement
- **Is auditing enabled?** Absence of logging means undetected intrusions and evidence-free attacks
- **Is data encrypted?** Unencrypted data at rest and in transit enables raw data theft from storage or backups
- **Is there a less monitored copy of this database?** Development, test, and backup environments often have weaker controls than production

Oracle Database's flexibility creates assessment complexity. Hardening requires understanding multiple interdependent factors:

- **User accounts, roles, and privileges**: Who accesses the database and what permissions they possess
- **Data classification and sensitivity**: What data resides in each database and its regulatory implications

- **Security parameters and features**: Which controls are configured and how they interact
- **Attack vectors relevant to the deployment**: Which threats matter most for the specific environment
- **Available prevention, detection, and response controls**: What security capabilities the organization can deploy

Implementing attacker-informed defenses

Using the attacker's playbook to guide defensive controls creates a more effective security posture:

Inventory and version hygiene: Maintain current versions and apply security patches on defined schedules. Knowing what databases exist and which versions they run prevents attackers from exploiting known vulnerabilities in outdated software.

Strong identity and access controls: Eliminate default accounts, enforce strong credential policies, and minimize privileges to essential functions. This reduces the attack surface by limiting what compromised credentials can accomplish.

Monitoring and auditing: Enable comprehensive auditing for sensitive operations and configure real-time alerts for anomalous behavior. Detection capabilities ensure attacks generate evidence and trigger investigation.

Encryption and key management: Encrypt data at rest and in transit with properly managed encryption keys and clear key ownership. Encryption renders stolen data useless without corresponding keys.

Environment mapping: Identify all database copies and non-production instances containing sensitive data, then align controls and auditing across all environments. Attackers target the weakest copy; consistent controls eliminate easy targets.

This approach enables rapid risk assessment, prioritization of high-impact remediations, and maintenance of a defensible security posture for Oracle Database deployments.

Fundamental considerations for database protection

Effective database protection requires understanding three fundamental realities that shape assessment priorities and remediation strategies:

Data sensitivity varies within and across databases. A customer's birth date typically carries more sensitivity than their email address due to identity theft implications. Not all tables within a database merit identical protection levels. Assessment must identify which databases contain sensitive data and classify it by type, including personally identifiable information (PII), protected health information (PHI), payment card data, and intellectual property. That classification directly drives risk-proportionate controls: which data gets encrypted, who can access it, and which audit policies apply.

Attack surfaces are multifaceted and interdependent. Vulnerabilities emerge from unpatched systems, weak passwords, excessive privileges, missing encryption, inadequate auditing, poor application design, and lack of separation of duties. Comprehensive assessment examines all these vectors simultaneously because attackers will exploit the weakest link regardless of other controls. A fully patched, encrypted database with weak password policies remains vulnerable to credential compromise.

Risk requires a business context. Not all databases warrant the same level of protection investment. Production systems holding regulated financial data demand stronger controls than development environments running synthetic test data. The important exception: any development system that contains a copy of production data requires production-equivalent controls, full stop.

Assessment tools should support risk-based prioritization that accounts for business criticality, data sensitivity, and regulatory obligations. Organizations should invest in security controls commensurate with what each database holds, covering the tools, personnel time, and operational resources that level of risk requires.

Users and accounts are diverse entities. Beyond database administrators, various actors and processes interact with data through database user accounts: applications, application administrators, security administrators, service accounts, batch programs, and increasingly, AI agents. Each user type has different legitimate access requirements. Assessment must identify these user types and their required activities to properly implement least privilege principles. Static privilege reviews miss the nuance of which privileges accounts actually use versus which they possess.

Configuration parameters have complex interdependencies. Security configuration parameters tightly relate to database behavior and require understanding their function, the impact of changes, and their dependencies on other parameters. Changing one parameter may inadvertently weaken another control or break application functionality. Assessment tools that explain parameter purposes and impacts enable confident remediation.

With this attacker-informed foundation, organizations can approach assessment tools strategically, understanding not just what they assess but why those assessments matter for defense.

Assessing configuration security with DBSAT

The Database Security Assessment Tool (DBSAT) provides the foundation for Oracle Database security assessment. DBSAT identifies where database configuration, operations, or implementation introduce risk, collecting and analyzing configuration data and parameters to deliver actionable remediation recommendations.

DBSAT assessment scope

DBSAT examines comprehensive security dimensions beyond basic database and listener configuration:

- **User accounts**: Account types, authentication methods, and account status
- **Privileges and roles**: Granted permissions, role hierarchies, and privilege paths
- **Authorization controls**: Access control mechanisms and enforcement policies
- **Separation of duties**: Administrative role segregation and least privilege implementation
- **Fine-grained access control**: Row-level security, column masking, and Virtual Private Database configurations
- **Data encryption and key management**: Transparent Data Encryption status, tablespace encryption, and key management practices
- **Auditing policies**: Audit configurations, audit trail security, and log retention
- **Operating system file permissions**: File ownership, directory permissions, and OS-level access controls

DBSAT scans for weaknesses and vulnerabilities and categorizes findings by risk level, including Pass, Low Risk, Medium Risk, High Risk, Advisory, and Evaluate, to support remediation prioritization. The tool provides high-level executive summaries alongside specific, actionable technical recommendations for each finding, helping teams move more quickly from insight to remediation.

DBSAT is a free command-line tool available to Oracle customers with current support agreements. Oracle consultants, partners, and external auditors around the world use DBSAT to perform standardized database security assessments. This broad adoption helps establish consistent assessment methodologies and a shared language for findings across the Oracle Database ecosystem.

DBSAT architecture and components

As seen in Figure 3-1, DBSAT consists of three discrete components serving different assessment purposes:

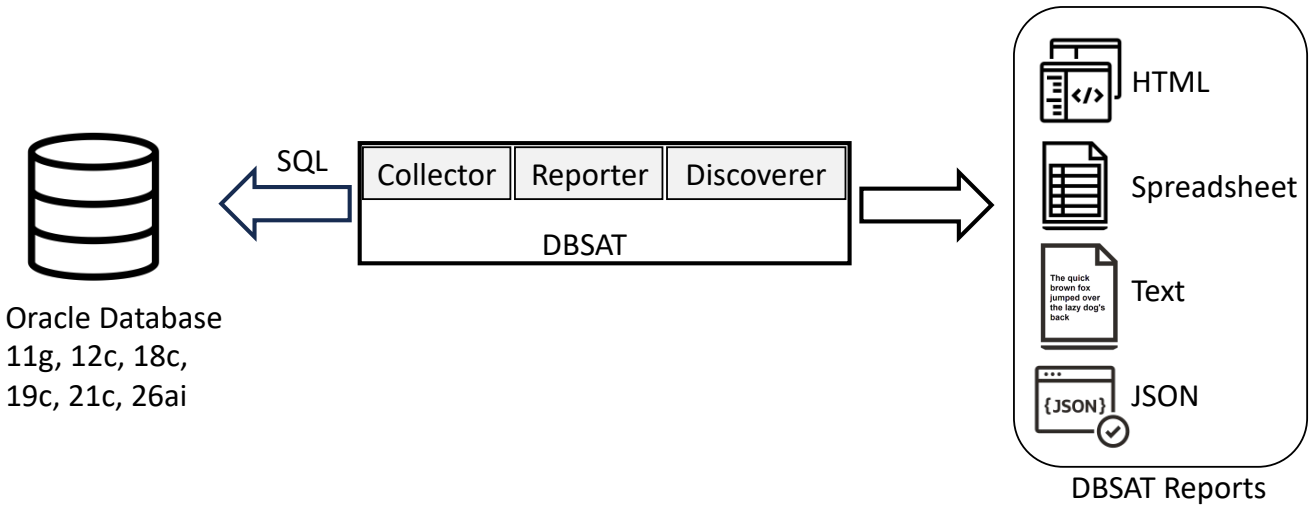


Figure 3-1: High-level functional diagram of Database Security Assessment Tool (DBSAT)

Collector: Gathers security configuration information from the database and underlying operating system. The collector connects to the database using the provided credentials, queries data dictionary views, reviews file system permissions, and creates a compressed JSON file containing the collected data. The collector does not perform analysis or make judgments. It captures the current state for later analysis.

Reporter: Analyzes the collected data and generates comprehensive reports with findings and recommendations. The reporter processes the collector's JSON output offline and compares configurations against Oracle best practices, CIS Benchmarks, DISA STIG requirements, and GDPR principles. This separation between collection and analysis enables organizations to collect data in production environments and perform analysis on separate systems.

Discoverer: Scans database schemas to identify sensitive data and classify it by type. The discoverer flags columns that contain sensitive information, including credit card numbers, Social Security numbers, email addresses, and phone numbers, and summarizes results in reports.

DBSAT report formats and usage

DBSAT generates reports in four formats, each serving specific use cases:

HTML format: Provides detailed, easily navigable assessment results with interactive summary dashboards. HTML reports are ideal for initial review, sharing with stakeholders, and understanding finding context. They include hyperlinked navigation between sections, expandable details, and embedded documentation references.

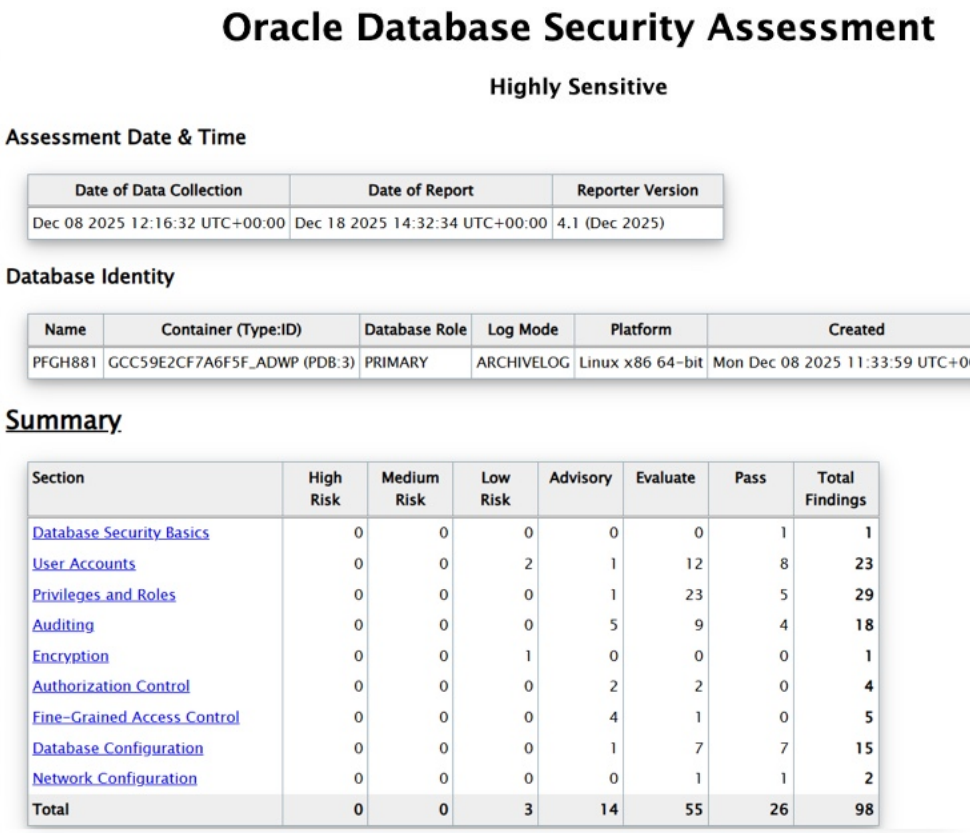
Spreadsheet format: Delivers high-level finding summaries in CSV format that loads into Excel or similar tools. Spreadsheets enable adding custom columns for tracking remediation status, assigning ownership, setting target dates, and prioritizing based on organizational risk tolerance. Many organizations use spreadsheet exports to integrate findings into existing issue tracking systems.

Text format: Outputs plain text suitable for scripting, version control, and command-line workflows. Text format facilitates copying portions of findings into documentation, tickets, or remediation procedures.

JSON format: Structured data ideal for programmatic processing, aggregation across multiple databases, and integration with security information and event management (SIEM) systems or governance, risk, and compliance (GRC) platforms.

Understanding DBSAT findings

Each DBSAT finding follows a consistent structure providing comprehensive context for security teams and DBAs. Figure 3-2 illustrates the DBSAT HTML summary report showing risk distribution across assessment categories.



Oracle Database Security Assessment

Highly Sensitive

Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Dec 08 2025 12:16:32 UTC+00:00	Dec 18 2025 14:32:34 UTC+00:00	4.1 (Dec 2025)

Database Identity

Name	Container (Type:ID)	Database Role	Log Mode	Platform	Created
PFGH881	GCC59E2CF7A6F5F_ADWP (PDB:3)	PRIMARY	ARCHIVELOG	Linux x86 64-bit	Mon Dec 08 2025 11:33:59 UTC+00:00

Summary

Section	High Risk	Medium Risk	Low Risk	Advisory	Evaluate	Pass	Total Findings
Database Security Basics	0	0	0	0	0	1	1
User Accounts	0	0	2	1	12	8	23
Privileges and Roles	0	0	0	1	23	5	29
Auditing	0	0	0	5	9	4	18
Encryption	0	0	1	0	0	0	1
Authorization Control	0	0	0	2	2	0	4
Fine-Grained Access Control	0	0	0	4	1	0	5
Database Configuration	0	0	0	1	7	7	15
Network Configuration	0	0	0	0	1	1	2
Total	**0**	**0**	**3**	**14**	**55**	**26**	**98**

Figure 3-2: Example DBSAT assessment summary table

The summary provides immediate risk visibility, which categories contain the most severe findings, total finding counts by risk level, and assessment metadata including database identity and collection timestamp.

Drilling into individual findings reveals detailed information structured to support rapid remediation. Figure 3-3 below shows a representative finding examining users with DBA role grants. Each finding contains:

Rule ID: A unique identifier with a prefix indicating the assessment category (e.g., PRIV for privileges, USER for user accounts, ENCRYPT for encryption) followed by a descriptive name. Rule IDs remain consistent across DBSAT versions, enabling tracking of specific findings over time and across databases.

One-liner: A concise sentence describing what the finding assesses. This summary enables quick scanning of multiple findings to identify priorities.

Status: Indicates risk severity or required action:

- **Pass**: Configuration meets security best practices
- **Low Risk**: Minor security concern with limited exploitation potential
- **Medium Risk**: Significant security concern requiring timely remediation
- **High Risk**: Critical security vulnerability demanding immediate attention

- **Advisory**: Information about optional security features not currently deployed
- **Evaluate**: Configuration requires manual review to determine risk level based on organizational context

Summary: Provides an overview of what the finding discovered in the assessed database. For example: "9 out of 53 users have been directly or indirectly granted highly sensitive DBA/PDB_DBA role via 9 grants."

Details: Presents specific findings from the database, listing affected users, configuration parameters, or security issues. In Figure 3-3, DBSAT identifies users with DBA roles, including SCOTT who received the role through nested role grants (SCOTT ← APPROLE1 ← APPROLE2 ← APPROLE3 ← DBA), a common obfuscation tactic.

Remarks: Explains why the finding matters and recommends specific remediation actions. For the DBA role finding, remarks emphasize limiting this powerful role to a small number of trusted administrators, creating custom DBA-like roles with minimum required privileges, and implementing Privilege Analysis to identify unnecessary permissions.

References: Maps findings to industry standards and compliance frameworks including CIS Oracle Database Benchmark recommendations, DISA STIG rules, and GDPR articles. This mapping helps organizations demonstrate compliance alignment and prioritize findings based on regulatory obligations.

Documentation: For Oracle Database 19c and later, provides direct hyperlinks to relevant Oracle documentation sections explaining parameters, features, and remediation procedures.

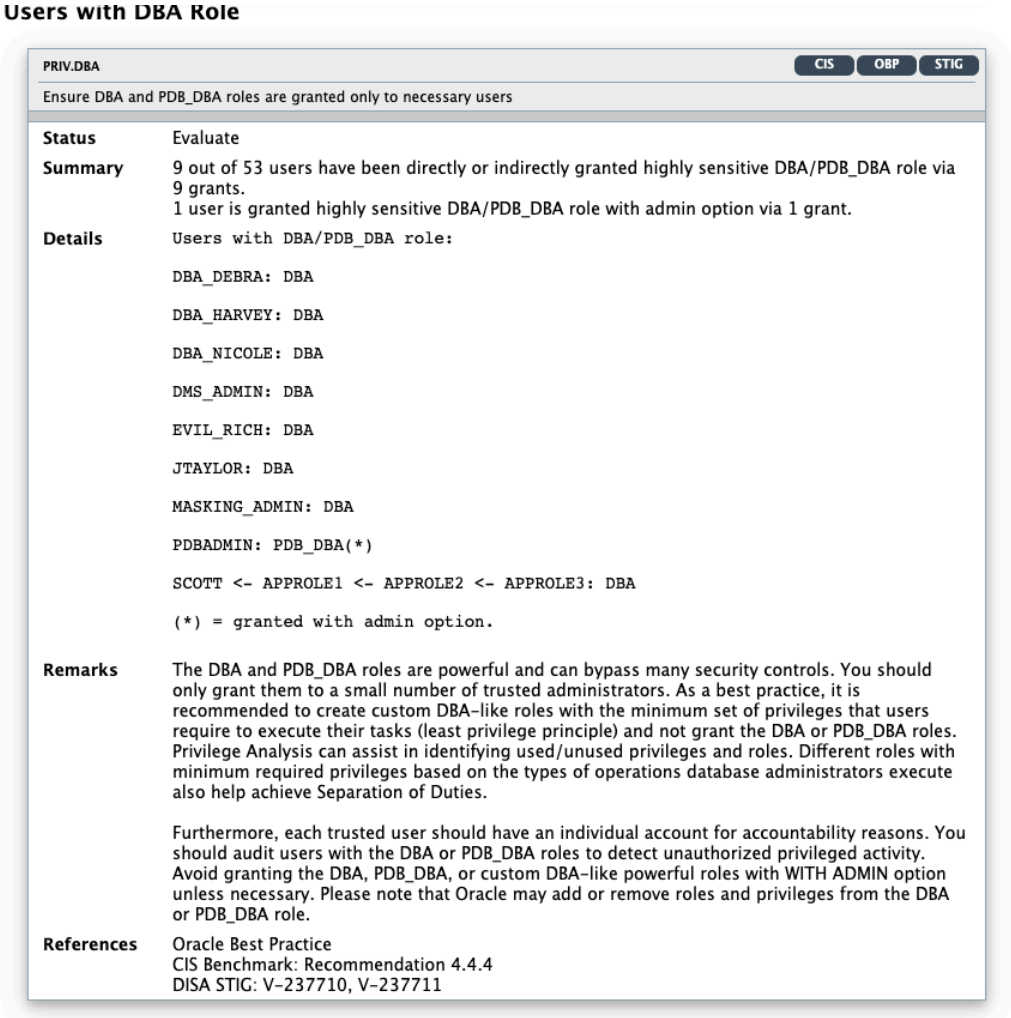


Figure 3-3: DBSAT Sample finding - Users with the DBA Role

In the illustrated finding, DBSAT identified eight users with the DBA role and one with PDB_DBA. Notably, SCOTT holds DBA privileges through a chain of nested role grants, a configuration that can be difficult to spot in a manual review but is surfaced through DBSAT's systematic analysis. The finding status, "Evaluate", signals that the DBA should review these grants to confirm they align with organizational needs. SCOTT may legitimately require DBA privileges as an application administrator, or the grants may reflect temporary access that was never removed and should have been revoked months ago.

This structured level of detail helps DBAs and security teams quickly understand what was found, why it matters, and what actions to take, even when team members do not have deep security expertise.

DBSAT deployment patterns

Organizations deploy DBSAT using several patterns depending on fleet size and operational maturity:

Manual execution for small deployments: Organizations with a handful of databases run DBSAT manually on a quarterly schedule, storing collector output in version control and tracking findings in spreadsheets. This approach works well for up to 1-10 databases with minimal automation investment.

Scripted execution for medium deployments: Organizations with dozens of databases automate collection using shell scripts that iterate through database inventories, execute the collector with stored credentials, and organize output by database name and timestamp. Analysis remains manual but collection automation ensures consistent assessment schedules.

Enterprise integration for large deployments: Organizations with hundreds or thousands of databases integrate DBSAT into existing automation frameworks (Ansible, Puppet, Chef), centralized configuration management databases, and security dashboards. Automated workflows collect data, execute the reporter, parse JSON output, and populate tracking systems. Exceptions and high-risk findings trigger automated tickets and notifications.

The tool's command-line nature and structured outputs make DBSAT amenable to extensive automation without requiring commercial orchestration platforms.

Assessing database fleets with Oracle Data Safe

While DBSAT excels at assessing individual databases, managing security across dozens, hundreds, or thousands of databases requires automation, centralized visibility, and fleet-wide orchestration. Manual assessment reviews do not scale. Oracle Data Safe addresses this challenge as an Oracle Cloud Infrastructure service purpose-built for Oracle Database security at scale.

Data Safe provides a unified view of fleet security posture, enabling security teams and DBAs to discover risk concentrations, prioritize remediation across multiple databases, and track security improvements over time. Data Safe works with Oracle Databases deployments regardless of location: on-premises data centers, Oracle Cloud Infrastructure, third-party clouds (AWS, Azure, Google Cloud), and Oracle Database@Multicloud configurations.

Data Safe capabilities beyond assessment

Data Safe delivers six integrated security capabilities within a single platform:

- **Security assessment**: Evaluates database configurations against best practices and compliance frameworks
- **User assessment**: Analyzes user accounts, privileges, roles, and risk profiles
- **Activity auditing**: Collects, consolidates, and analyzes database audit trails
- **Security policy management**: Deploys and enforces consistent security configurations across fleets
- **SQL Firewall**: Monitors and blocks unauthorized SQL statements
- **Data discovery and masking**: Identifies sensitive data and anonymizes non-production environments

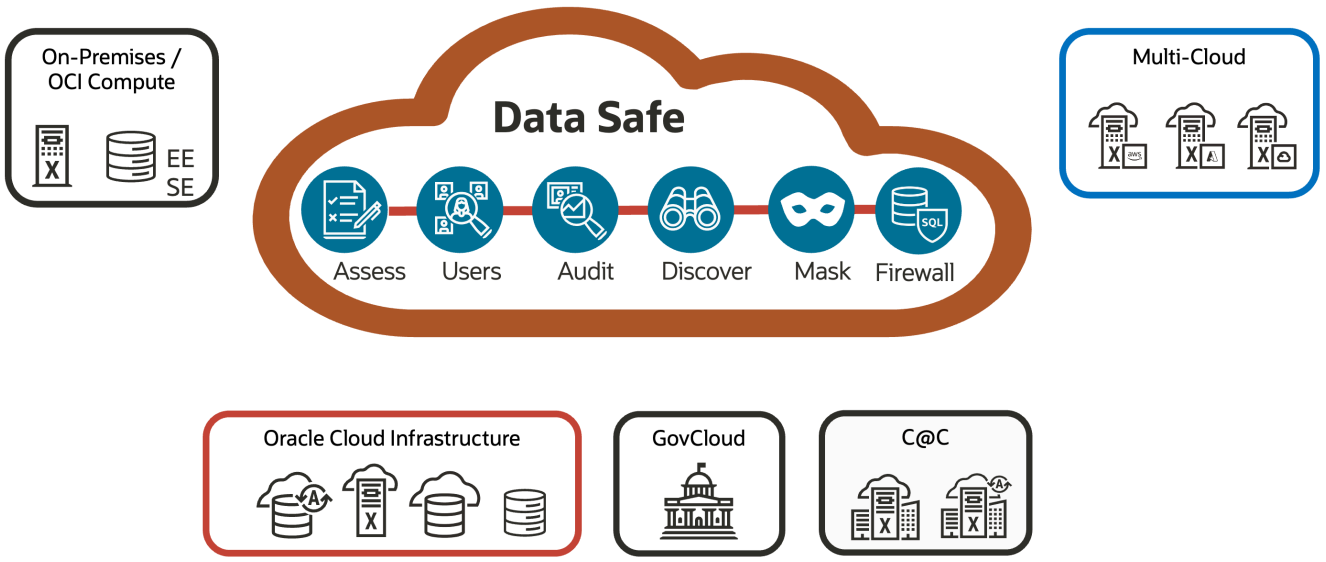


Figure 3-4: Oracle Data Safe provides essential security for Oracle AI Databases, both in the cloud and on-premises

Data Safe security assessment

Data Safe security assessment performs comprehensive configuration evaluation identical in scope to DBSAT but with enterprise features enabling fleet management:

Assessment areas evaluated:

- User accounts and authentication configurations
- Privilege and role grants
- Authorization controls
- Fine-grained access control policies
- Auditing policies and configurations
- Encryption status and key management
- Database configuration parameters
- Network security and access controls

Data Safe compares discovered configurations against organizational best practices and flags deviations. Reports include prioritized recommendations mapped to compliance frameworks including GDPR, DISA STIG, and CIS Benchmarks, ensuring remediation efforts align with regulatory obligations and industry standards.

Fleet-Scale features

Data Safe enhances DBSAT's assessment capabilities with features essential for managing large database populations:

Simultaneous multi-database assessment: Launch assessments across all fleet databases simultaneously rather than sequentially assessing one at a time. Parallel execution completes fleet-wide assessments in minutes regardless of database count.

Scheduled recurring assessments: Define assessment schedules (weekly, monthly, quarterly) that execute automatically. Scheduled assessments ensure continuous monitoring without manual intervention and establish trend data showing security posture over time.

Security baselines: Designate an assessment as the security baseline representing the desired configuration state. Subsequent assessments compare against this baseline to identify configuration drift. Baselines enable tracking what changed rather than re-reviewing all findings each cycle.

Drift reports: Generate reports highlighting differences between baseline and current assessments. Drift reports show which databases diverged from approved configurations, which new findings appeared, and which previous findings were remediated. This differential view focuses remediation efforts on actual changes.

Assessment history: Retain historical assessments to analyze security posture trends. History enables answering questions like "Are we improving?" and "When did this configuration drift occur?"

Fleet aggregation: Consolidate findings across all databases to identify systemic issues. If 80% of databases have identical misconfigurations, the problem likely originates from deployment templates or operational procedures requiring correction.

Filtering and search: Query findings across the entire fleet using database name, risk level, finding type, or custom tags. Large organizations can quickly identify all databases with specific high-risk findings requiring immediate attention.

Event-driven notifications: Configure alerts that trigger when assessments discover High Risk findings or when configuration drift exceeds thresholds. Notifications integrate with email, ticketing systems, and incident management platforms.

Figure 3-5 shows the Data Safe assessment summary interface providing immediate visibility into fleet risk distribution. Unlike DBSAT's static HTML report, Data Safe provides an interactive dashboard enabling drill-down from summary metrics into specific findings and affected databases.

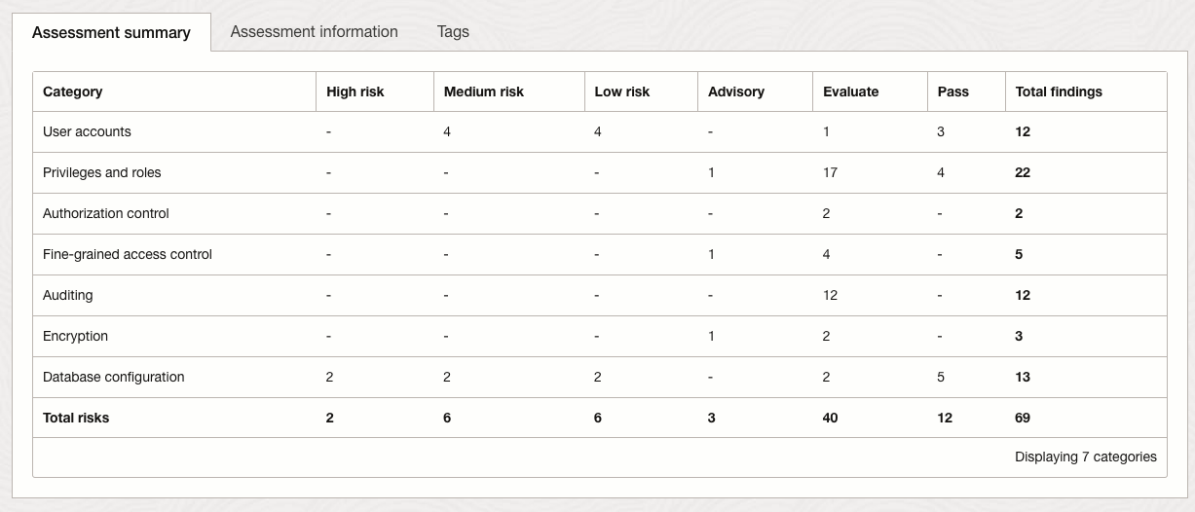


Figure 3-5: Example Data Safe assessment summary

Data Safe finding management

Data Safe utilizes the same assessment rules as DBSAT but enhances finding management with enterprise workflows. Figure 3-6 illustrates detailed findings with risk categorization and compliance framework mapping.

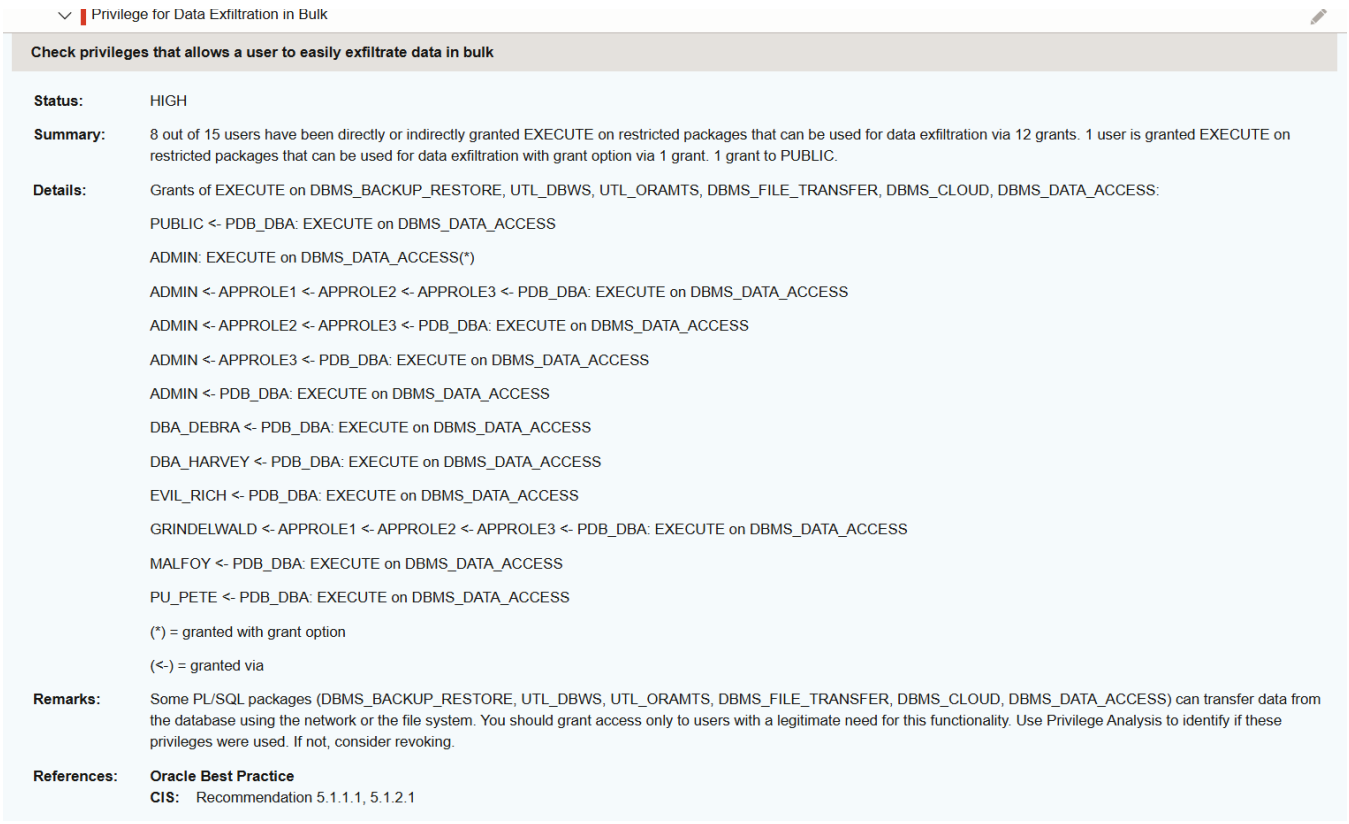


Figure 3-6: Example Data Safe security assessment findings

Data Safe also enables configuration of acceptable risks. For example, when an assessment flags a finding for evaluation (Status EVALUATE, as shown in Figure 3-7: Example), the Data Safe user is expected to review it and determine whether it represents a real issue. In the provided example, Data Safe identifies two users, ADMIN and SKEETER, with the privilege to manage audit settings, but the reviewer must decide whether SKEETER's privilege is actually necessary.

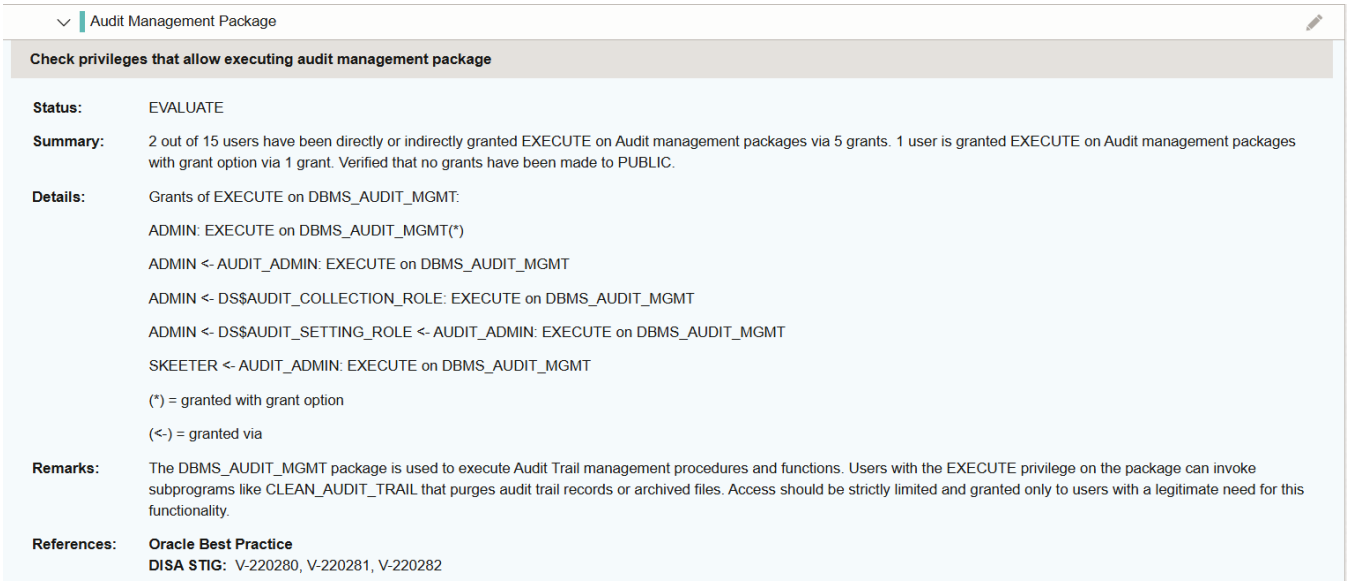


Figure 3-7: Example Data Safe finding requiring evaluation

If a review confirms the finding presents no risk (i.e., SKEETER legitimately requires the privilege), you can mark the finding as “Pass.” You can also lower a finding’s risk if compensating controls are in place. Alternatively, you can choose to accept the risk if no remediation is planned. In the scenario illustrated, SKEETER’s privilege was validated, and Data Safe was configured to remind the user to review the finding again later.

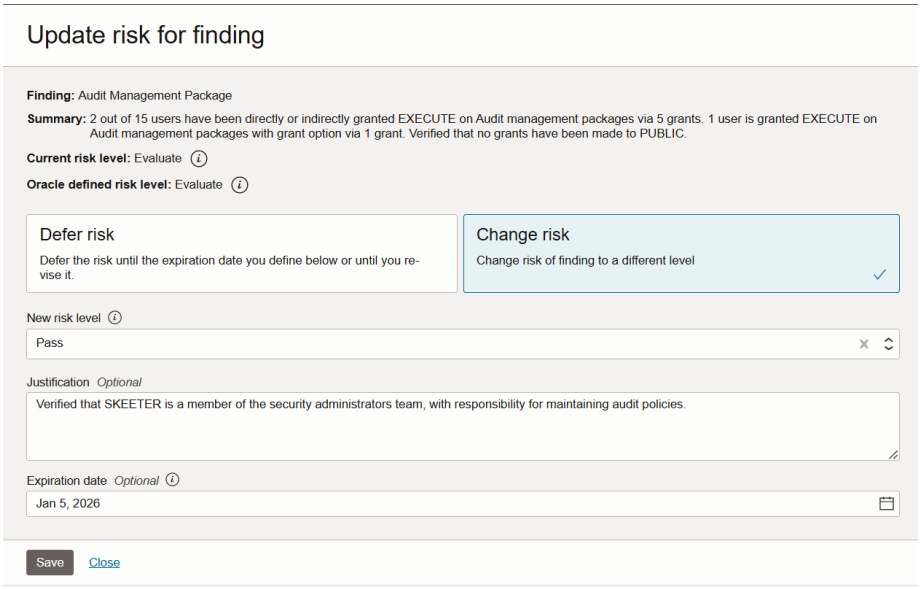


Figure 3-8: Example Data Safe screen for configuring acceptable risks

Updated risks are easy to identify, since any changes from the default values are clearly shown in Figure 3-9.

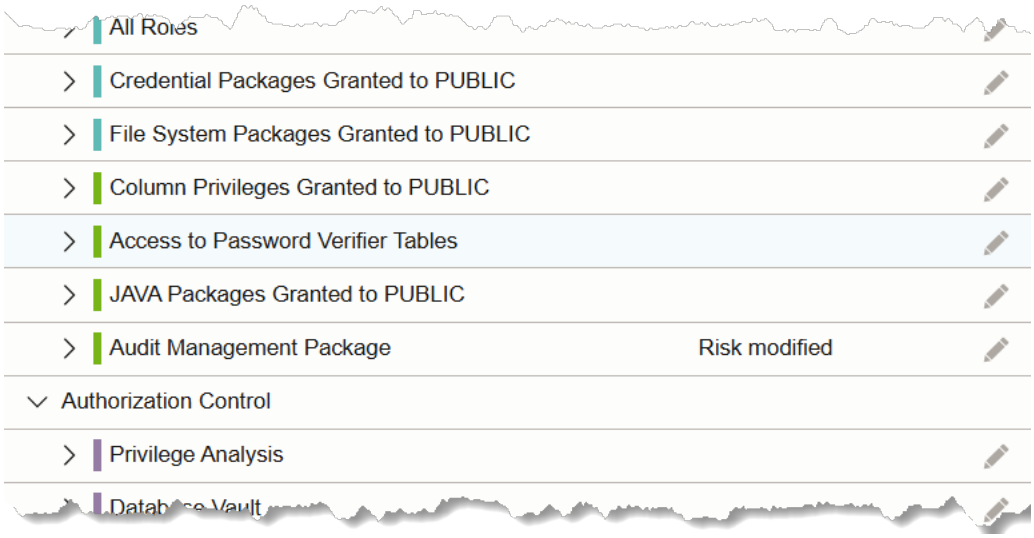


Figure 3-9: Example screen for finding modified risks

And security assessment's *risk modification report* lets you quickly drill into details on the modification.

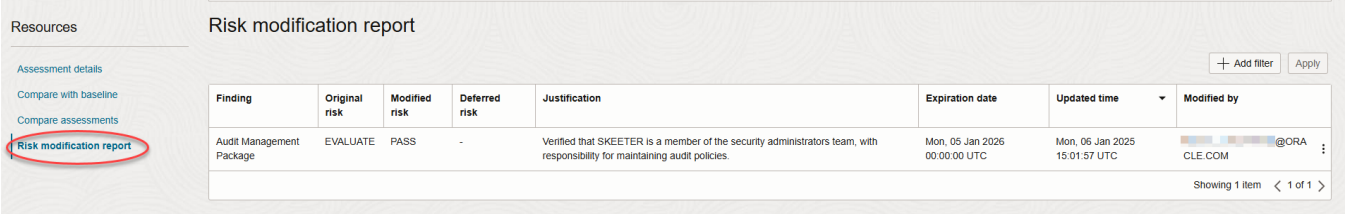


Figure 3-10: Example Security Assessment risk modification report

Understanding user risk with Data Safe

Privileged credential theft ranks among the most common attack vectors enabling data breaches. Data Safe user assessment helps mitigate this risk by evaluating database accounts and visualizing user risk to enable rapid intervention.

User assessment analyzes accounts using both static and dynamic characteristics:

Static profile analysis: Assigned roles, granted privileges, profile settings, authentication methods, and account creation date

Dynamic behavioral analysis: Last login timestamp, password change recency, failed login attempts, and database session patterns

The assessment highlights highest-risk users based on privilege levels and suspicious patterns. Figure 3-11 illustrates the user risk dashboard showing potential risk distribution, role assignments, and password change patterns.

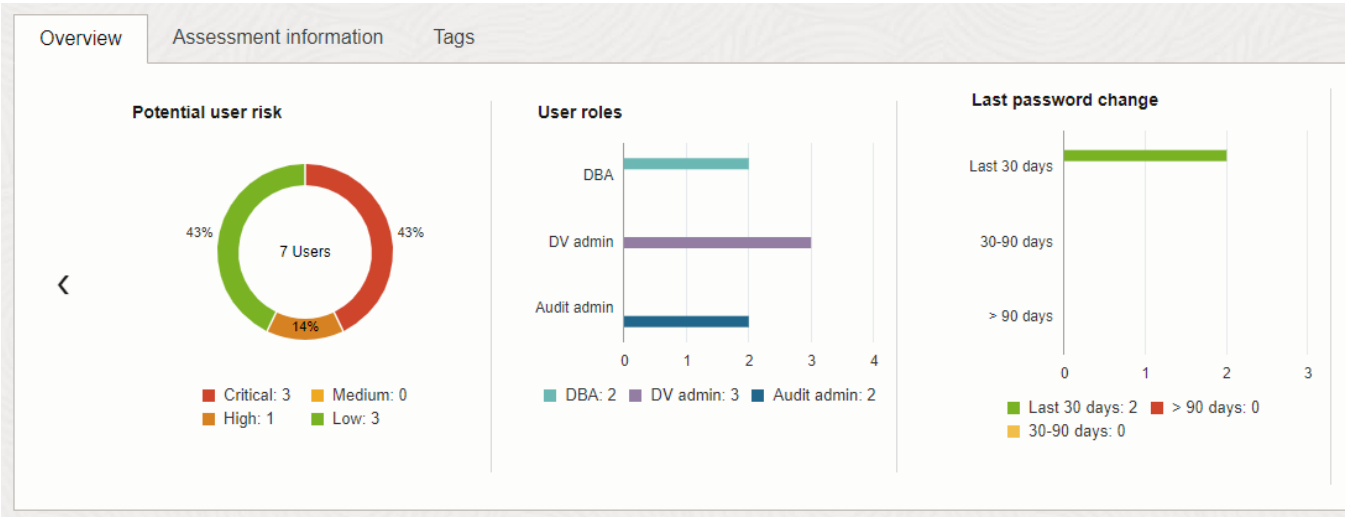


Figure 3-11: Example Data Safe user risk assessment

Security teams can quickly identify concerning patterns:

- Users who have not authenticated in three months or longer (stale accounts potentially forgotten)
- Users who have not changed passwords in 90+ days
- High-privilege accounts (DBA, SYSDBA) without corresponding audit trails
- Service accounts with interactive login capabilities

When suspicious activity is detected, Data Safe enables drilling into account details: creation date, assigned roles and privileges, privilege grant paths: and opening related audit reports to review recent actions. This integrated approach connects static privilege analysis with runtime behavior analysis.

User profile management

Database user profiles control authentication security, password policies, and resource limits. Profiles represent critical security controls but are often inconsistently deployed across database fleets. Data Safe's User Profile Insight provides visibility into profile configurations across all registered databases.

Organizations can:

- Review existing profiles and parameters: Examine password verification functions enforcing complexity, failed login attempt limits, and password expiration policies
- Compare profiles across databases: Identify differences between identically named profiles on different databases indicating configuration drift
- Identify profile-to-user assignments: Understand which users are protected by which profiles
- Find weak configurations: Quickly locate users and profiles with inadequate password governance

Figure 3-12 shows User Profile Insight summarizing profile distribution and highlighting users without proper password complexity requirements.

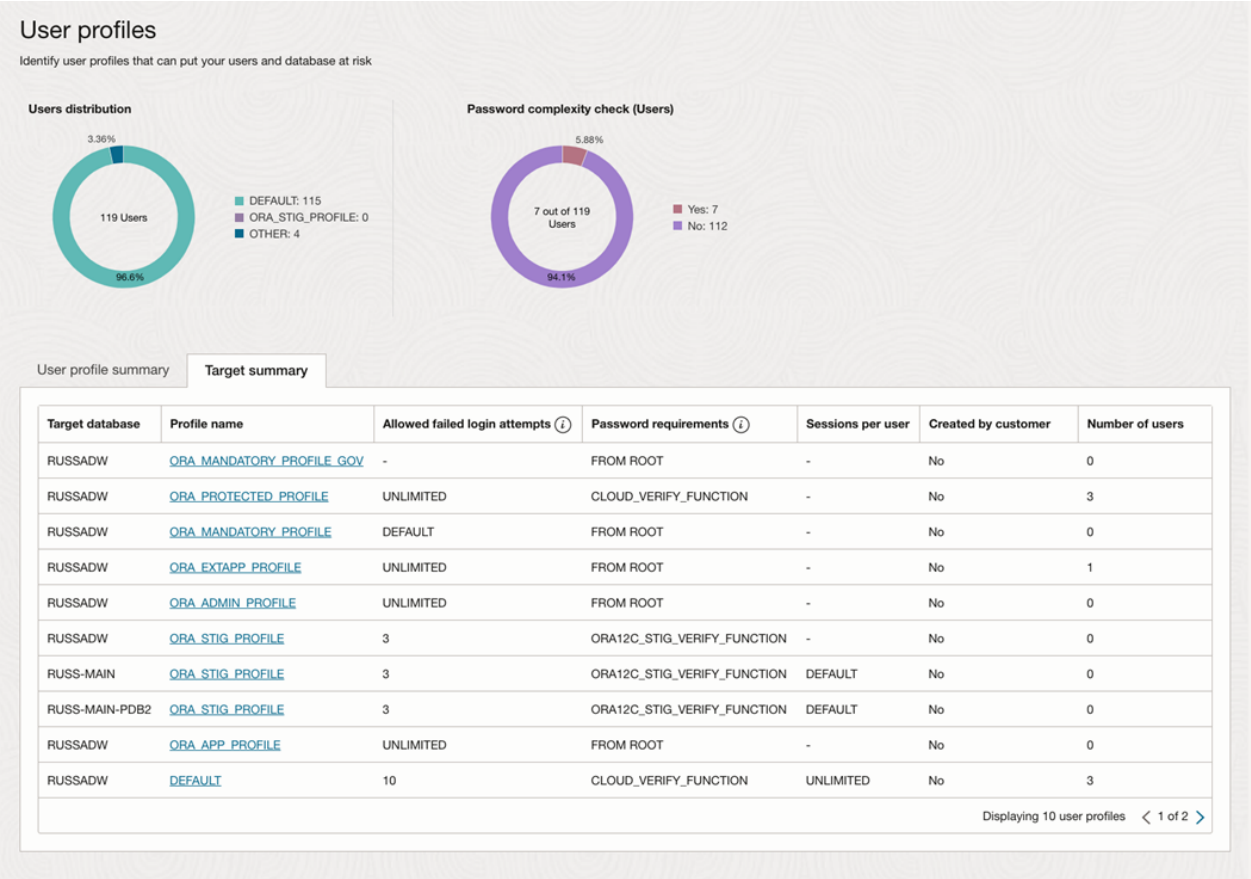


Figure 3-12: Example Data Safe user profile insight

Data Safe deployment and architecture

Data Safe operates as a multi-tenant Oracle Cloud Infrastructure service. Organizations deploy Data Safe Targets: registered databases: in one of several configurations:

Cloud-native databases: Oracle Autonomous Database, Base Database Service, and Exadata Cloud Service instances register with Data Safe automatically during provisioning. No additional configuration is required.

On-premises databases: Traditional on-premises Oracle Database instances connect to Data Safe through Oracle Data Safe Private Endpoints that establish secure network tunnels between on-premises networks and Oracle Cloud Infrastructure. Private Endpoints enable Data Safe functionality without exposing databases to the public internet.

Third-party cloud databases: Oracle Database instances running on AWS, Azure, or Google Cloud connect through Private Endpoints or network peering configurations.

Oracle Database@Multicloud: Oracle Database services running in multi-cloud environments register natively through cloud-to-cloud network connectivity.

Data Safe's cloud-based architecture eliminates infrastructure management overhead: no servers to maintain, no software to patch, no capacity planning required. Oracle manages the service infrastructure, ensuring availability, scalability, and security.

Organizations with regulatory requirements prohibiting cloud-based management services can use Oracle Database Security Central (covered in the next section) which provides similar assessment capabilities with on-premises deployment.

Having explored Data Safe's assessment capabilities, the next section examines Oracle Database Security Central's security posture management features for organizations requiring on-premises solutions.

Assessment with Oracle Database Security Central (Security Central)

Security Central provides comprehensive database security posture management for organizations preferring or requiring on-premises deployment. While Data Safe operates as a cloud service, Security Central deploys as a software appliance in customer data centers, providing similar assessment capabilities with on-premises data residency.

The Security Insights in **Security Central** Console provides a unified, actionable view of your organization's database security risks by delivering an in-depth assessment of security posture across your Oracle Database fleet. It analyzes key areas such as configurations, user accounts, and sensitive data to surface potential risks.

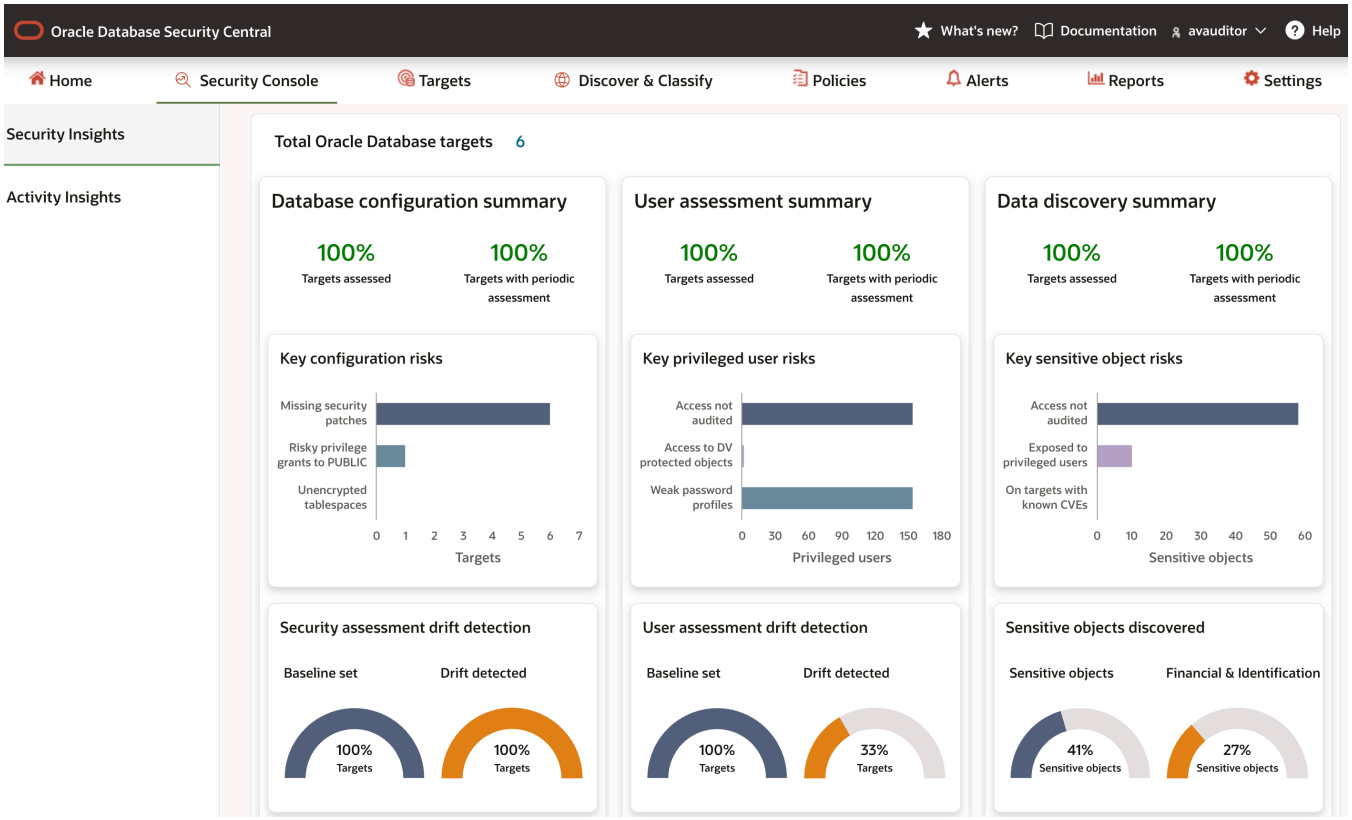


Figure 3-13: Example Security Insights posture management dashboard

By correlating insights across the database fleet into a simplified operational view, Security Insights enables security teams to quickly identify high-risk areas, prioritize remediation efforts, and take focused action to strengthen overall security posture and reduce organizational risk.

Security Insights organizes risk visibility around three integrated assessment pillars. Each pillar addresses a distinct dimension of database security risk, and together they provide a comprehensive understanding of security posture across the database estate.

Assessment Pillar	Focus Area	Key Capabilities
User 360	Privileged Access and Identity Risk	Discovers users, roles, and privileges across the fleet to identify high-risk access, detect dormant or excessive privileges, and track entitlement drift over time.
Data 360	Sensitive Data Discovery and Classification	Identifies and classifies sensitive data across production, test, and development environments using predefined and customizable sensitive data types to help align monitoring, auditing, and access controls with actual data sensitivity.
Configuration 360	Security Posture Management	Continuously evaluates database configurations against approved baselines and industry standards such as CIS and DISA STIG to detect configuration drift, prioritize vulnerabilities, and maintain consistent security posture across the fleet.

Table 3-1: Assessment pillars in Security Central

The three pillars deliver the greatest value when their insights are correlated. By combining user risk, sensitive data exposure, and configuration posture into a unified view, Security Insights helps organizations identify compound risks, prioritize remediation based on overall impact, and accelerate investigations with a more complete understanding of database security risk.

User 360: Privileged Access and Identity Risk

Privileged accounts remain one of the most significant sources of database security risk. In large environments, access privileges often accumulate through direct grants, nested roles, inherited permissions, and temporary exceptions, making it difficult to understand who truly has elevated access and what level of risk that access introduces. Dormant accounts, stale passwords, excessive privileges, and entitlement drift can create hidden exposure paths that are difficult to detect through isolated reviews.

Figure 3-14 shows Security Central's user assessment summary displaying fleet-wide risk distribution

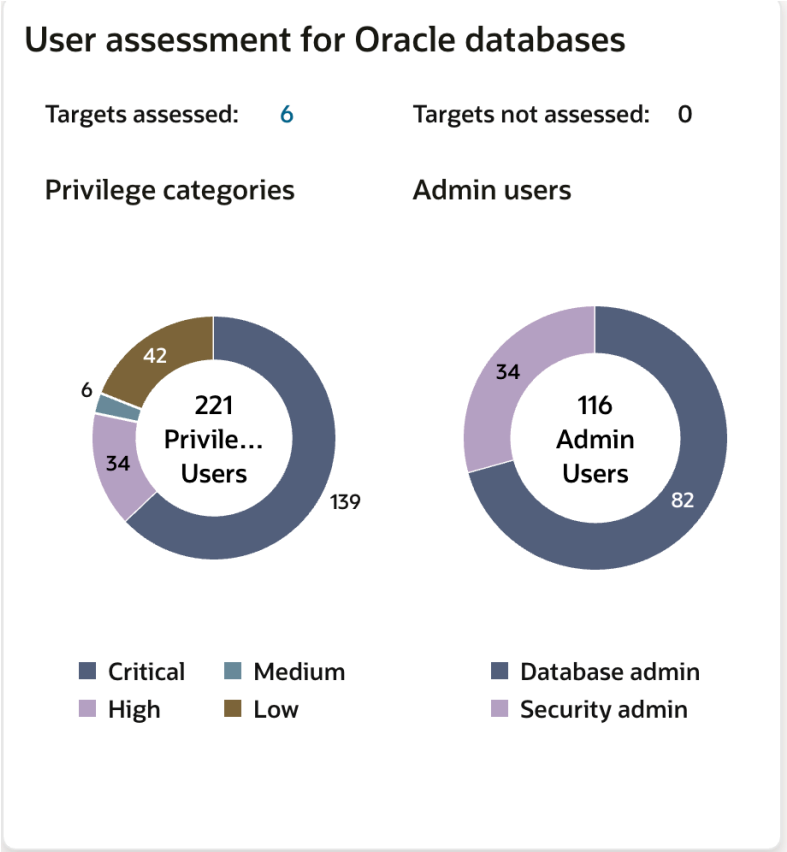


Figure 3-14: Example Security Central user assessment summary

The dashboard provides immediate answers to critical questions: How many databases are assessed? How many are pending user assessment? What is the overall user risk distribution?

User 360 provides fleet-wide visibility into database users, roles, privileges, and entitlements across the Oracle Database estate. It maps both direct and indirect privilege relationships to reveal effective access paths, identifies highly privileged and potentially risky users, and assigns automated risk scores based on privilege levels and account characteristics. The solution also helps security teams detect dormant accounts, weak password policies, expired or locked accounts, and excessive privilege assignments that may increase organizational risk.

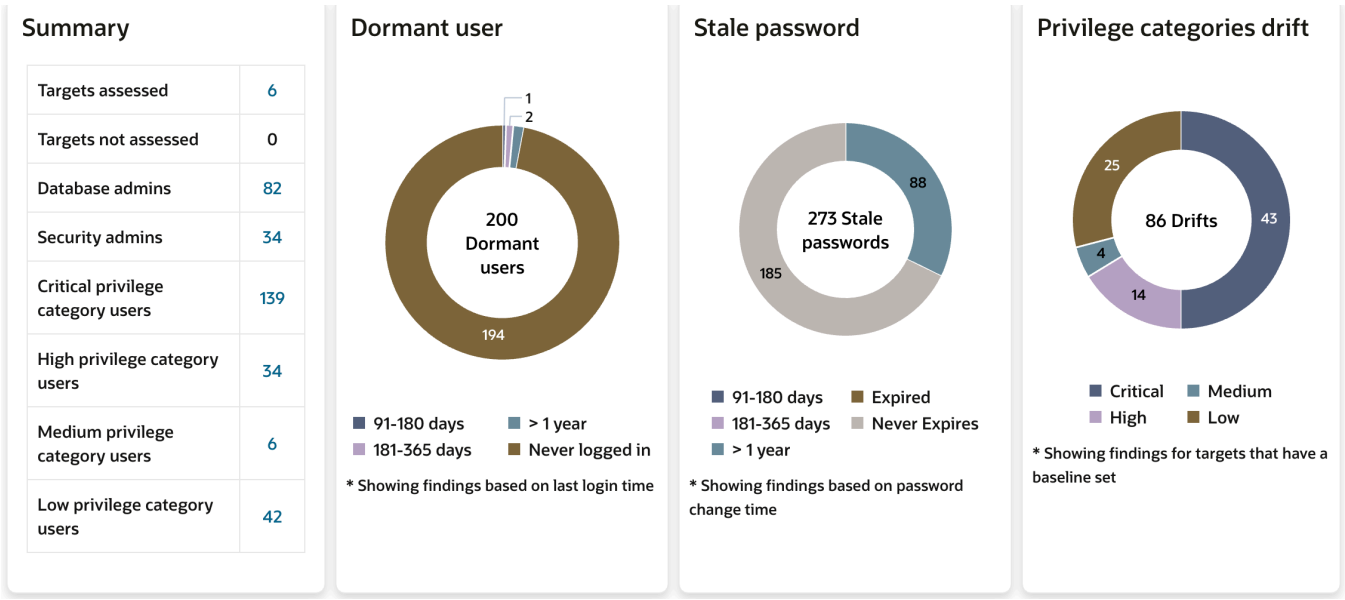


Figure 3-15: Example Security Central user assessment reviews

Comparative auditing in User 360 as shown in Figure 3-16 enables security teams to identify differences in user access, roles, and privileges across databases and historical snapshots, helping detect entitlement drift, unauthorized changes, and inconsistent privilege assignments over time.

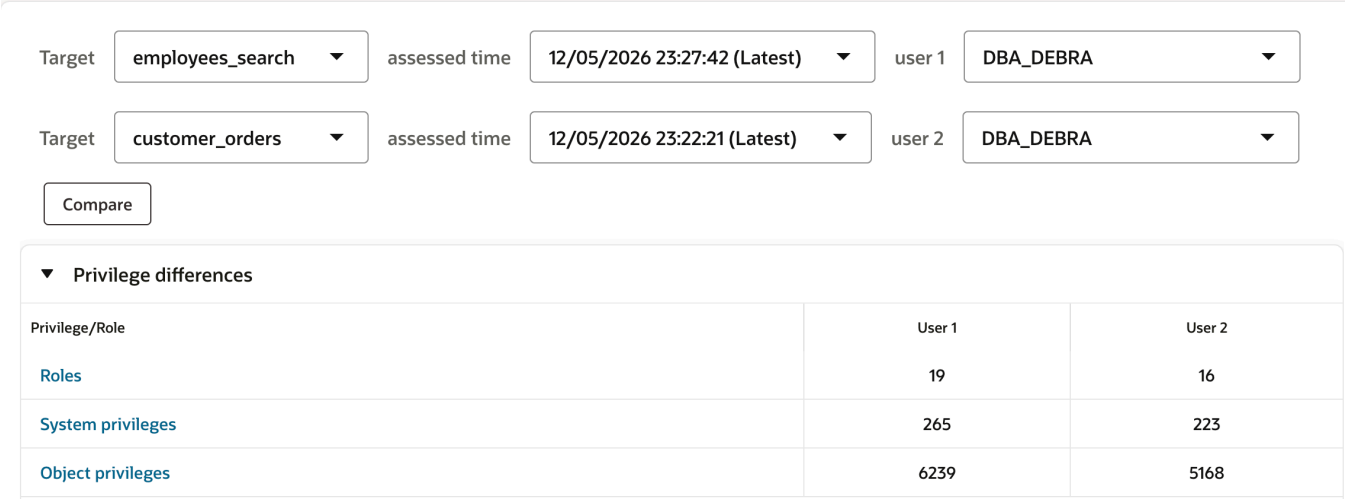


Figure 3-16: Example user comparative auditing (DBA_DEBRA access across two different instances)

By continuously monitoring privileged access and user activity, User 360 provides fleet-wide user risk intelligence that helps organizations identify high-risk accounts, detect entitlement drift, and uncover excessive or dormant privileges across the database estate. Through centralized visibility, automated risk assessment, and comparative auditing, security teams can enforce least-privilege principles, prioritize access reviews, and reduce the risk of privilege misuse or unauthorized access before those risks are exploited.

Data 360: Sensitive Data Discovery and Classification

Sensitive data is often distributed across production, test, development, and reporting environments, making it difficult for organizations to maintain consistent visibility and protection. Without an accurate understanding of where sensitive data resides and how it is classified, security teams face challenges in applying the right controls, monitoring access, and prioritizing protection efforts.

Data 360 provides fleet-wide visibility into sensitive data across the Oracle Database estate by discovering, classifying, and quantifying sensitive information using more than 180 predefined sensitive data types across 20 categories. It combines metadata analysis with row-level data validation to improve discovery accuracy, supports custom sensitive data types and classification groups, and enables scheduled assessments to maintain consistent classification as data evolves across the environment.

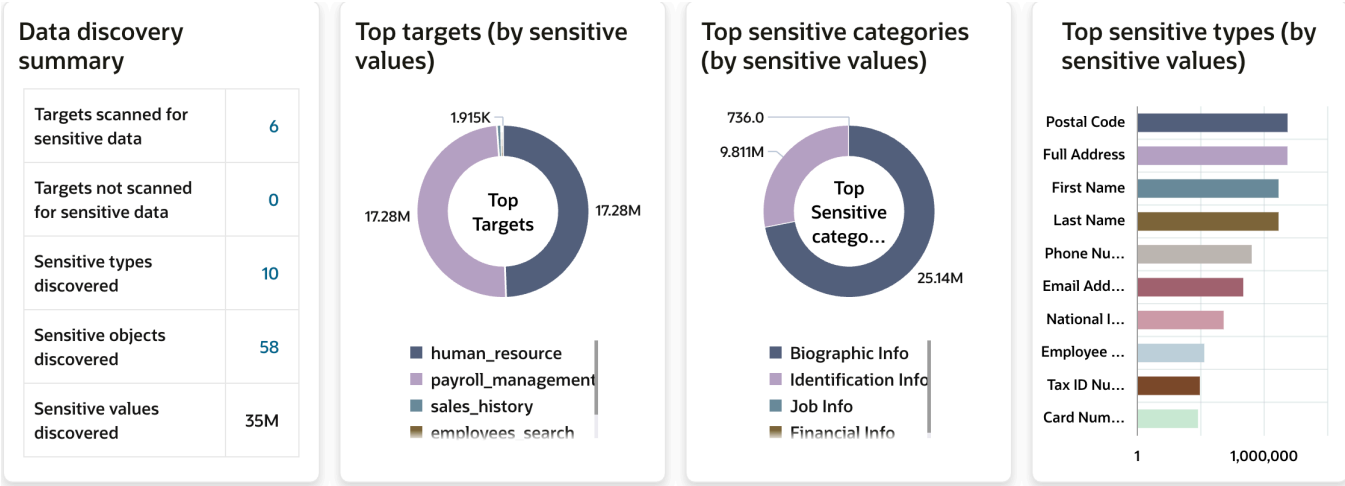


Figure 3-17: Example Security Central sensitive data discovery dashboard

By delivering fleet-wide sensitive data intelligence through centralized dashboards and reporting, Data 360 helps organizations identify high-risk data exposure, align monitoring and access controls with actual data sensitivity, and prioritize protection efforts across the database estate.

Configuration 360: Security Posture Management

Security Central delivers fleet-wide centralized assessment with compliance framework mapping and actionable recommendations. Organizations establish security baselines, schedule recurring assessments, and generate drift reports showing how current configurations diverge from approved baselines. This differential approach enables focusing on actual changes rather than re-reviewing all findings with each assessment cycle.

Configuration 360 provides continuous fleet-wide assessment of database security posture by evaluating database configurations against approved baselines and industry standards such as CIS and DISA STIG. It detects configuration drift, prioritizes findings based on severity and exposure, maps assessments to regulatory frameworks including GDPR, and supports scheduled assessments to maintain continuous audit readiness and compliance visibility. Security Central's assessment engine leverages the same assessment rules as DBSAT and Data Safe, ensuring consistent finding definitions and risk classifications across Oracle's assessment tools. Figure 3-13 shows Security Central's security assessment summary displaying fleet-wide risk distribution

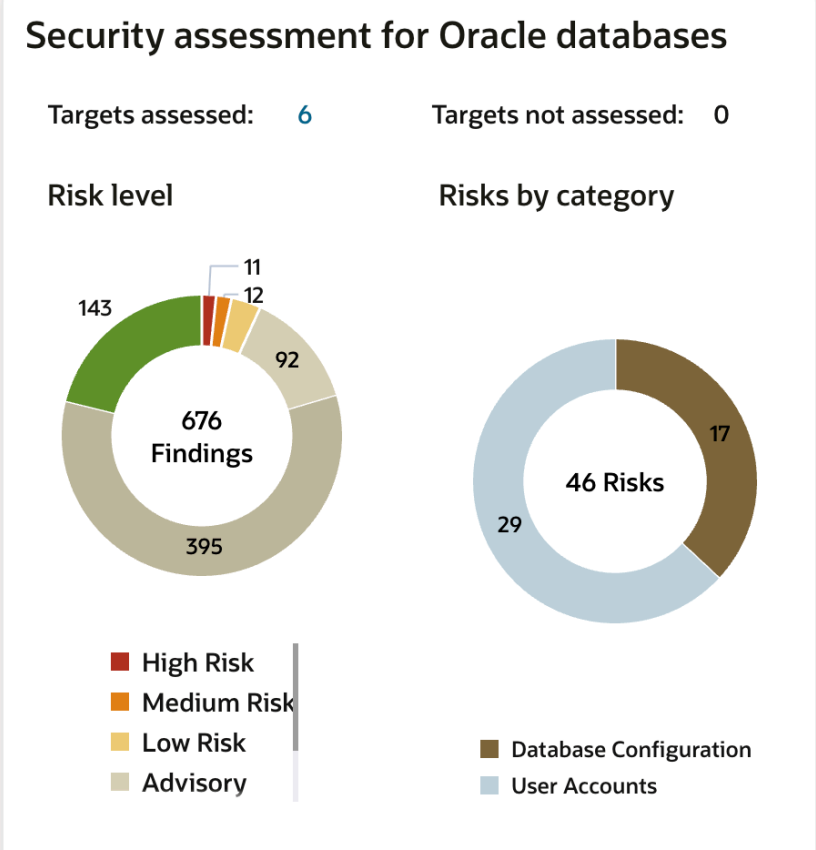


Figure 3-18: Example Security Central security posture management dashboard

The dashboard provides immediate answers to critical questions: How many databases are assessed? How many are pending security assessment? What is the overall risk distribution? Which categories contain the most findings?

As highlighted in the Security Insights console in Figure 3-19, configuration 360 findings such as missing security patches provide organizations with actionable visibility into critical configuration risks that require immediate remediation. Missing security patches remain one of the most significant indicators of database security exposure, especially as AI-enabled attack techniques can accelerate vulnerability discovery and exploitation by rapidly identifying weaknesses, reverse-engineering patches, and chaining vulnerabilities across systems to launch large-scale attacks.

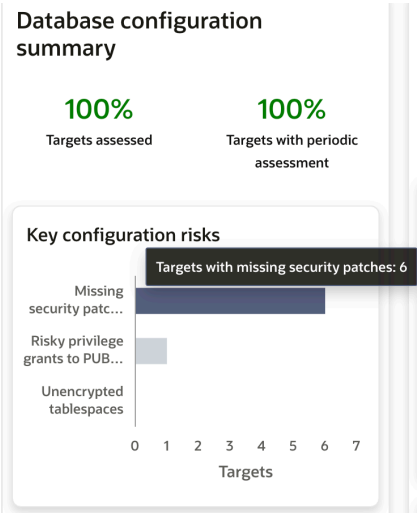


Figure 3-19: Example Missing security patches providing actionable visibility

Detailed security configuration assessment reporting

Security Central's reporting engine leverages Oracle Application Express (APEX), providing interactive drill-down navigation to precise information. Figure 3-20 displays the detailed assessment report showing all targets that require immediate security patch updates.

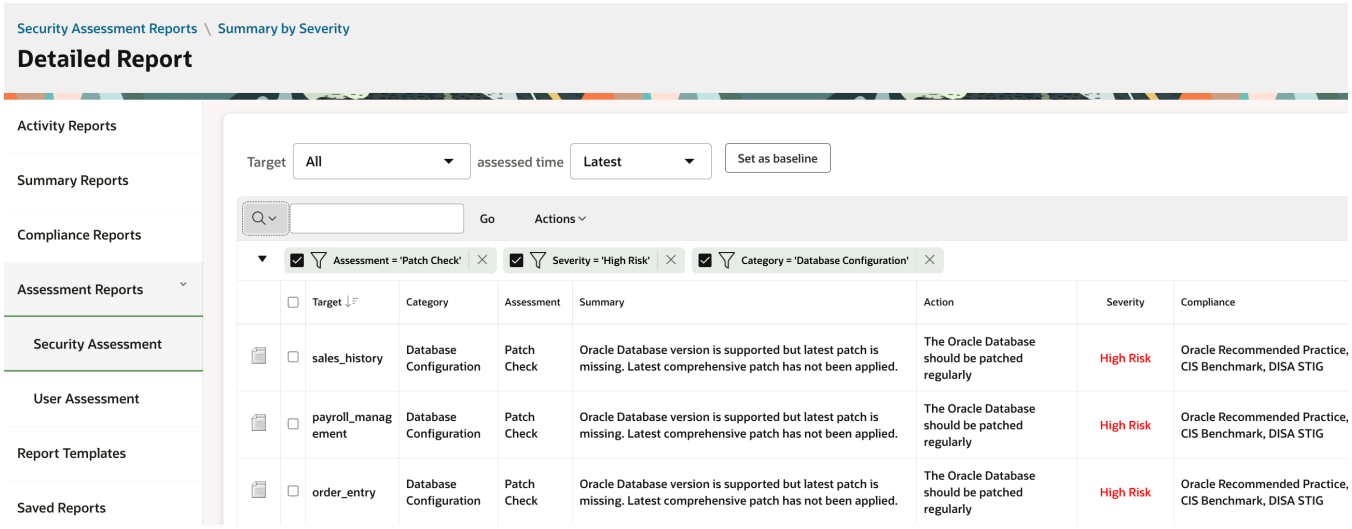


Figure 3-20: Example Security Central detailed assessment report for missing security patches

Organizations can filter findings by:

- Target database
- Risk level (High, Medium, Low, Advisory, Evaluate)
- Assessment category (User Accounts, Privileges, Encryption, Auditing)
- Compliance framework (CIS, STIG, GDPR)

From summary views, users drill into individual findings to access complete remediation guidance. Figure 3-21 illustrates detailed information for missing security patch finding with granular details.

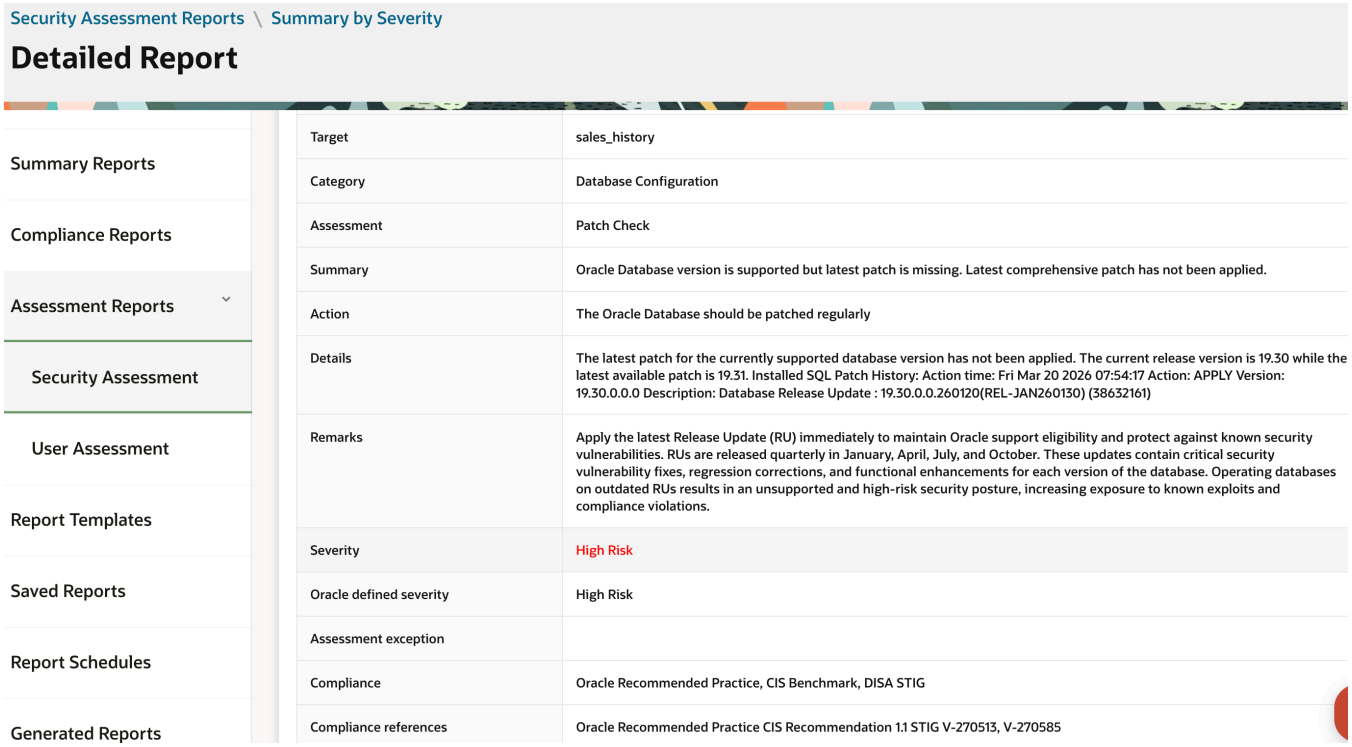


Figure 3-21: Example Security Central detailed finding on missing security patch

By delivering centralized posture visibility, baseline monitoring, actionable remediation insights, and automated assessments, Configuration 360 helps organizations reduce configuration-related risk while maintaining continuous compliance and audit readiness across the database estate.

Security Central beyond security posture management

Similar to Data Safe, Security Central provides capabilities beyond security posture management. Organizations use Security Central for:

- **Database activity monitoring**: Real-time monitoring and alerting on suspicious database activity
- **Centralized audit management**: Collecting, securing, and analyzing audit trails from multiple databases
- **Security policy management:** Centralized policy management for audit, DB Firewall, SQL Firewall, Database Vault, and alerts
- **Network monitoring:** Inspects SQL traffic in real time to detect or block unauthorized access and SQL injection threats before they reach the database with Database Firewall

Security Central's integrated platform approach combines assessment, monitoring, and audit management in a single appliance, reducing tool sprawl and simplifying operations.

With Security Central covering on-premises deployment scenarios, the next section examines Oracle Enterprise Manager's Database Lifecycle Management Pack for organizations with existing Enterprise Manager investments.

Configuration assessment with EM Database Lifecycle Management

Oracle Database Lifecycle Management Pack (DBLM) for Enterprise Manager automates database security assessment and compliance monitoring for organizations with existing Enterprise Manager deployments. DBLM integrates assessment into the broader database lifecycle management framework encompassing patching, provisioning, configuration management, and performance monitoring.

DBLM Compliance Framework

DBLM provides predefined compliance standards and customizable frameworks with detailed reports identifying configuration deviations. This enables maintaining secure, consistent configurations across database estates while reducing manual review time.

The compliance dashboard, Figure 3- below, delivers centralized visibility into compliance status across all managed databases. It presents compliance scores, highlights non-compliant targets, and enables drill-down into detailed reports for prioritization and resolution.

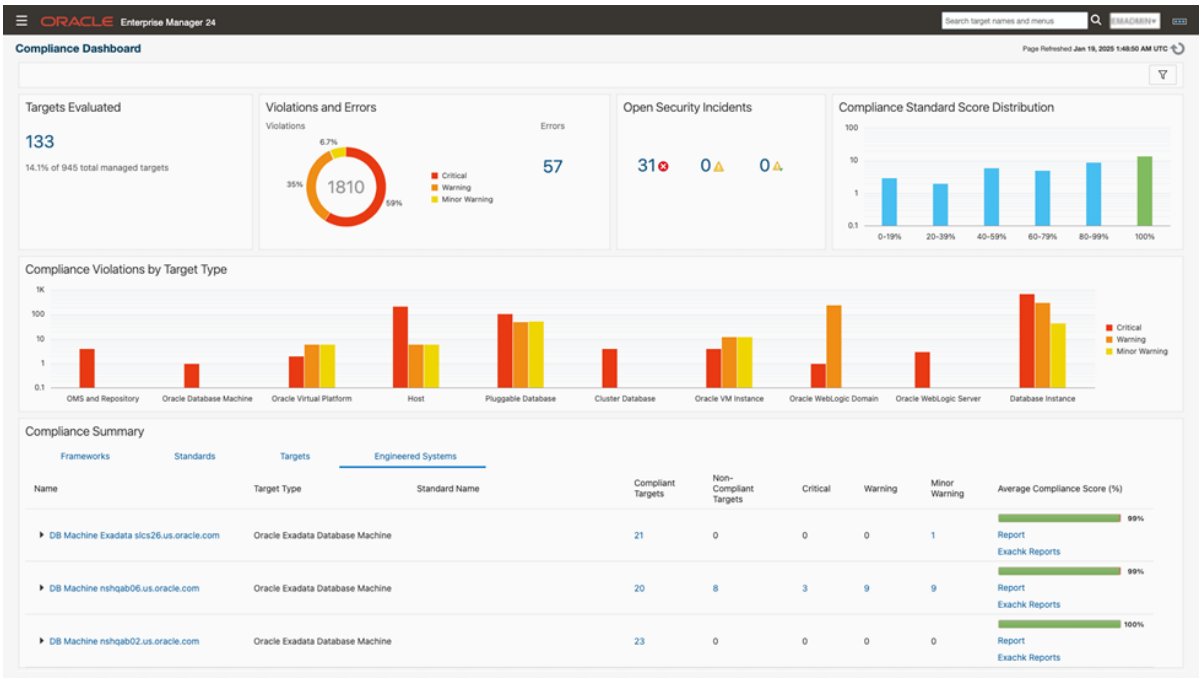


Figure 3-23: Example Oracle Enterprise Manager compliance dashboard

DBLM's compliance frameworks address multiple security dimensions:

- **Initialization parameters**: Security-relevant parameter configurations
- **Operating system directory permissions**: File ownership and access controls
- **User account profiles**: Password policies and resource limits
- **User access and authorization**: Privilege grants and role assignments
- **Auditing configuration**: Audit policies and audit trail security
- **Sensitive objects**: Access controls for critical database objects

Figure 3- shows the general security reports library providing quick access to common security assessments.

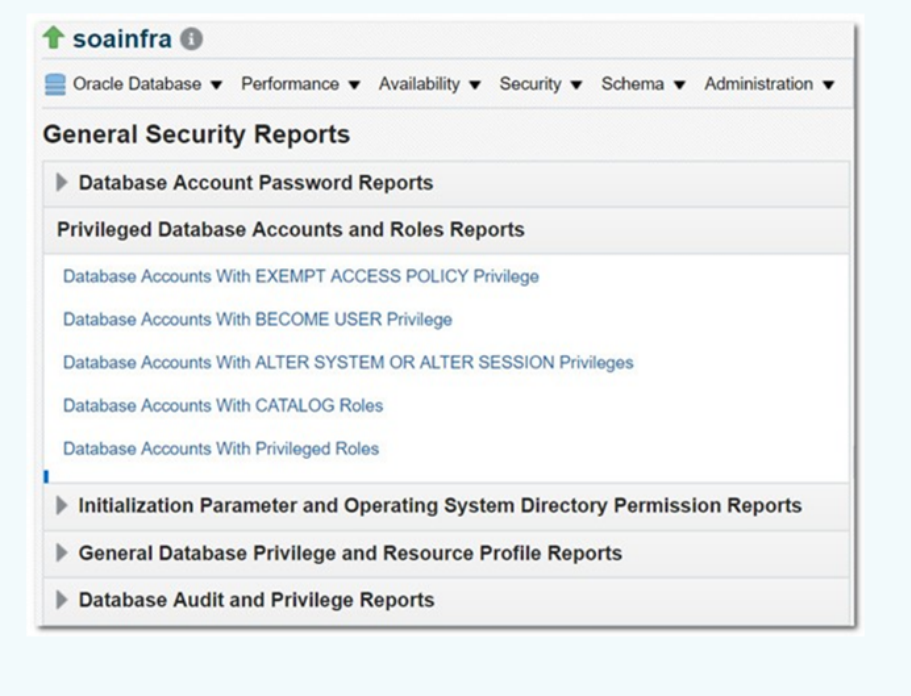


Figure 3-24: Example Enterprise Manager general security reports

Standards-based compliance monitoring

DBLM incorporates predefined compliance standards including:

- CIS Benchmarks: Center for Internet Security Oracle Database Benchmarks
- DISA STIG: Defense Information Systems Agency Security Technical Implementation Guide
- Oracle Database Security Best Practices: Oracle-recommended configurations

Organizations apply these standards across database fleets, examine detailed reports, and rapidly identify remediation priorities. Figure 3- illustrates compliance results for the STIG standard showing target-specific compliance scores and violation details.

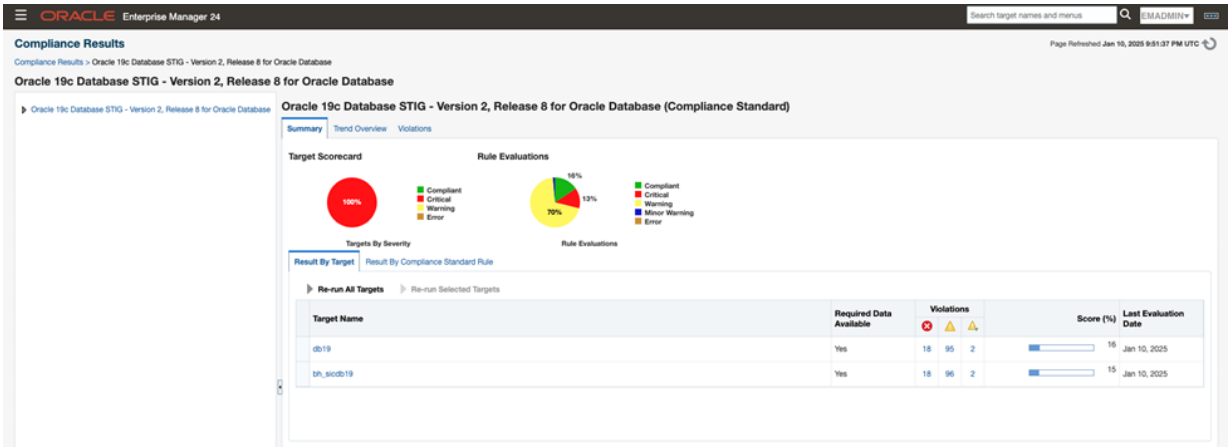


Figure 3-25: Example Enterprise Manager STIG compliance results

DBLM's standards include detailed rules validating specific configurations:

- Oracle Database initialization parameters

- Oracle Real Application Clusters (RAC) node configurations
- Oracle Listener security settings
- Operating system security for database server hosts

Organizations can customize standards, create proprietary frameworks, and define organization-specific requirements. Custom standards enable encoding internal security policies as automated compliance checks.

Oracle Database Security Assessment Tool (DBSAT) can also be run from within Oracle Enterprise Manager to automate assessments and centralize operations. This approach enables organizations to:

- Schedule and run DBSAT assessments alongside other management tasks
- Consolidate findings for improved fleet-wide visibility
- Continuously identify, assess, and mitigate risk across databases

Oracle Autonomous Health Framework (AHF) integrates with compliance workflows to strengthen Exadata security by running automated configuration checks and health checks. When managed through Enterprise Manager, organizations gain:

- Centralized oversight of Exadata security posture
- Continuous monitoring and detailed reporting to speed remediation
- Proactive risk mitigation and alignment with regulatory requirements for secure, compliant Exadata environments

Managing configuration drift and consistency with Enterprise Manager

When configurations drift, outages and audit findings often follow. DBLM includes configuration management capabilities designed to keep database configurations consistent and compliant at scale, reducing risk and accelerating issue resolution. DBLM helps organizations:

- Create and maintain approved configuration baselines
- Continuously detect configuration drift and identify what changed, where, and when
- Enforce compliance standards and deploy automated remediation
- Produce detailed reports for audits and operational reviews

DBLM integrates with patching workflows to track expected changes and isolate unexpected drift. It also sends real-time alerts, enabling teams to investigate and resolve issues before they affect availability or security.

By automating monitoring and remediation, DBLM reduces manual effort, improves system reliability, and supports regulatory compliance. Organizations gain clearer audit trails, faster reviews, and lower risk, all managed from a centralized console that provides a consistent view across environments.

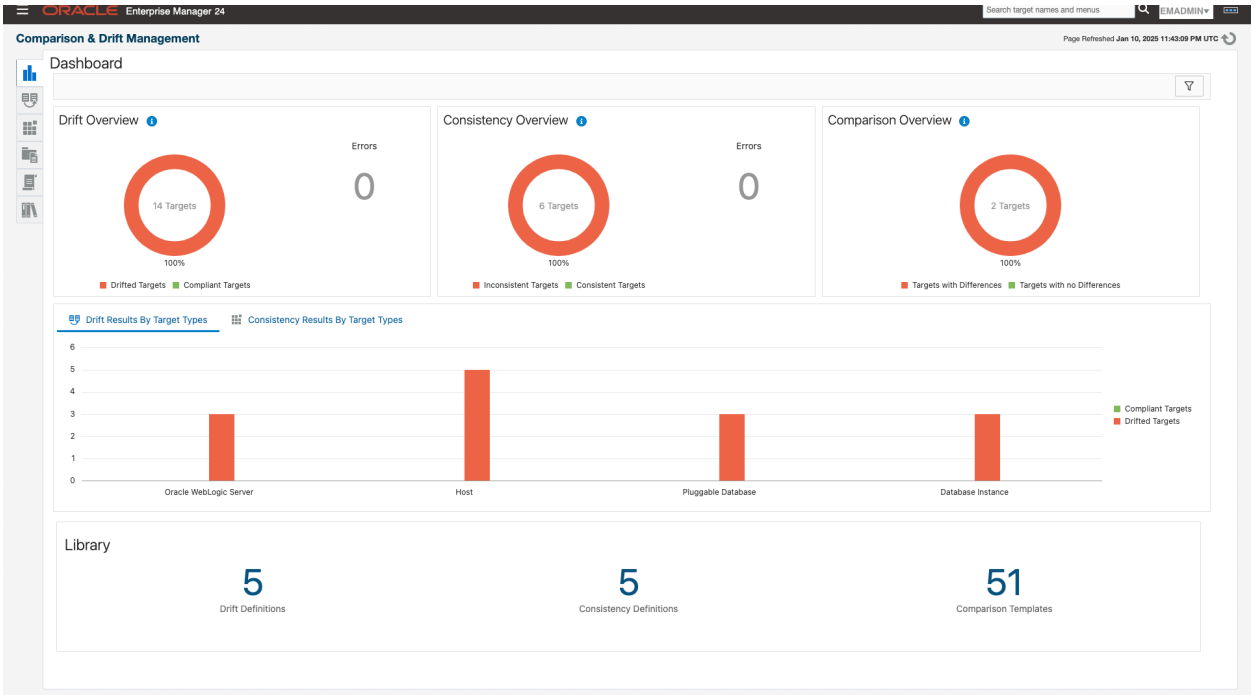


Figure 3-26: Example configuration drift and consistency management dashboard

Discovering and grouping assets

Manual IT asset tracking does not scale. DBLM combines asset inventory and usage tracking to deliver a unified inventory for hardware, databases, middleware, and applications, showing what you own, where it operates, and how you use it. DBLM automates:

- Automatic asset discovery and inventory maintenance
- Version tracking, configurations, and interdependencies
- Resource usage monitoring and licensing agreement alignment

Real time estate visibility supports confident capacity planning, cost optimization, and performance tuning. Lineage and dependency data helps evaluate change impact before action is taken. DBLM works with Oracle Enterprise Manager lifecycle tools to streamline patching, provisioning, and configuration. Organizations can:

- Coordinate patching and configuration changes with accurate inventory data
- Use current usage and configuration details to prioritize remediation
- Generate audit and license review reports within minutes

By automating discovery, monitoring, and remediation, organizations reduce manual effort, improve reliability, and stay compliant, enabling faster decisions and lower risk through a centralized console with consistent cross environment visibility.

Selecting the optimal Database Security assessment tool

Organizations have four primary tools for assessing Oracle Database security: DBSAT, Data Safe, Security Central, and DBLM. Selecting the optimal tool depends on fleet size, deployment architecture, existing tool investments, and operational constraints.

The following decision framework helps match organizational requirements to tool capabilities:

Database Security Assessment Tool (DBSAT)

Use DBSAT when organizations need straightforward, standalone assessments of individual Oracle Databases without infrastructure investment. DBSAT works well for:

- **Small deployments (1-10 databases)**: Manual quarterly execution with spreadsheet tracking
- **Proof-of-concept assessments**: Understanding Oracle Database security posture before investing in enterprise tools
- **Audit preparation**: Generating comprehensive findings for external audit evidence
- **Disconnected environments**: Air-gapped systems without internet connectivity

If organizations need to assess fleets, monitor configuration drift over time, or automate recurring assessments, transition to Data Safe, Security Central, or DBLM for fleet-scale capabilities.

Oracle Data Safe

Select Data Safe when organizations need security assessments at scale across numerous databases with cloud-based management. Data Safe provides the best fit for:

- **Medium to large deployments (10-1000+ databases)**: Automated fleet-wide assessment and centralized visibility
- **Multi-cloud and hybrid architectures**: Unified management for databases regardless of location (on-premises, Oracle Cloud, AWS, Azure, Google Cloud)
- **Organizations embracing cloud operations**: Preference for SaaS over on-premises infrastructure
- **Integrated security platforms**: Need for assessment alongside activity auditing, SQL Firewall, data discovery, and masking

Data Safe eliminates assessment infrastructure management overhead: no servers to maintain, patch, or capacity plan. Oracle manages service availability, scalability, and security.

Oracle Database Security Central (Security Central)

Choose Security Central when organizations require on-premises assessment capabilities at scale or when regulatory constraints prohibit cloud-based management services. Security Central suits:

- **Large on-premises deployments**: Hundreds to thousands of databases in customer data centers
- **Regulatory data residency requirements**: Assessment data must remain in specific geographic locations or on-premises
- **Integrated security platforms (on-premises)**: Need for security risk assessment, database activity monitoring, audit management, and unified policy management
- **Stored procedure integrity monitoring**: Detecting backdoor injection into stored procedures
- **Heterogeneous database environment:** Unified, enterprise-wide database security platform that standardizes compliance and threat mitigation across heterogeneous environments, including Oracle, Microsoft SQL Server, MySQL, PostgreSQL, and IBM Db2.

Security Central provides capabilities comparable to Data Safe but with on-premises deployment. Organizations preferring Data Safe's cloud benefits but constrained by regulatory requirements should carefully review those requirements: many regulations allow cloud services with proper contractual protections.

Oracle Enterprise Manager Database Lifecycle Management Pack

Select DBLM when organizations already extensively use Oracle Enterprise Manager for database management and do not require separation of database administration and security responsibilities. DBLM fits:

- **Existing Enterprise Manager deployments**: Organizations already managing databases through Enterprise Manager
- **Integrated lifecycle management**: Assessment alongside patching, provisioning, configuration management, and monitoring
- **Combined DBA and security roles**: Smaller organizations where DBAs perform security functions
- **Exadata environments**: Leveraging AHF integration for infrastructure security monitoring

DBLM reduces tool sprawl by consolidating assessment into existing management infrastructure. However, organizations with mature security teams preferring dedicated security tools may find Data Safe or Security Central provide better role separation.

Multitool strategies

Organizations need not choose exclusively. Many deploy multiple tools serving different purposes:

- DBSAT for ad-hoc assessments + Data Safe for continuous fleet monitoring
- DBLM for infrastructure configuration monitoring + Data Safe for security-specific assessments
- DBSAT for air-gapped systems + Data Safe for cloud and connected systems

The consistent finding definitions across tools enable comparing results regardless of which tool performed the assessment.

After reviewing configuration assessment tools, the next section turns to privilege analysis, focusing on not only which privileges are granted, but also which privileges are used.

Minimizing the blast radius with privilege analysis

Any security assessment seeks to reduce risk, which means looking beyond configuration settings. A significant share of database risk stems from user accounts and their privileges. Attackers often steal credentials to authenticate and extract data. Removing unnecessary privileges reduces the potential blast radius when an account is compromised.

Determining whether an account has excessive access can be difficult. Organizations can perform entitlement reviews, where a knowledgeable reviewer examines each user's roles and privileges and identifies what to remove. Tools like Data Safe user assessment and Security Central entitlement reviews can help, but managers may worry about disrupting operations, and DBAs may hesitate to revoke access when the consequences are unclear.

A more effective approach is to monitor which privileges accounts actually use and remove the rest. Oracle Database privilege analysis (PA) tracks privilege and role usage for database users and application service accounts over time. PA reports unused privileges and roles based on observed activity, enabling teams to remove unused access with confidence instead of relying on a manager's memory or guesswork. This data driven approach can reduce risk faster while minimizing operational friction.

PA reports show the privileges and roles that users and applications have used and not used. The `DBA_USED_PRIVS` and `DBA_UNUSED_PRIVS` views indicate which privileges and roles have been exercised, and which have not.

Figure 3-21 shows that the `APPS` user has privileges that are not being used: `DROP ANY TABLE`, `ALTER ANY TABLE`, `CREATE TABLE`, and `UNLIMITED TABLESPACE`. Additionally, `DROP ANY PROCEDURE` and `CREATE PROCEDURE` granted through both the APPS and `APPS_PATCHING` roles were not in use. Could this indicate someone granted the `APPS` user a role once for patching and never revoked it?

S/N	Policy	Grantee	Grantee Type	System Privileges	Grant Path
1	HR Analysis Policy	APPS	USER	DROP ANY TABLE	APPS
2	HR Analysis Policy	APPS	USER	ALTER ANY TABLE	APPS
3	HR Analysis Policy	APPS	USER	CREATE TABLE	APPS
4	HR Analysis Policy	APPS	USER	UNLIMITED TABLESPACE	APPS
5	HR Analysis Policy	APPS	USER	DROP ANY PROCEDURE	APPS,APPS_PATCHING
6	HR Analysis Policy	APPS	USER	CREATE PROCEDURE	APPS,APPS_PATCHING

Figure 3-27: Example report on DBA_UNUSED_PRIVS

Privilege analysis enables deeper investigation. Figure 3-22 reveals interesting details about used roles and privileges concerning specific database objects. Here, we observe that someone granted the APPS user `SELECT ANY TABLE` when, in fact, the user is selecting from specific tables on the HR schema (`DEPARTMENTS`, `JOB_HISTORY`, `COUNTRIES`, `EMPLOYEES`, `LOCATIONS`, `REGIONS`, and `JOBS`). In this scenario, the APPS account's inherent risk can be reduced by revoking the `SELECT ANY TABLE` system privilege, which allows the APPS user to query any table in the database, and replacing it with object privilege grants on only the required tables (for example, `GRANT SELECT` on `HR.DEPARTMENTS` to APPS). If APPS requires access to all tables in the HR schema, `GRANT SELECT ANY TABLE` on HR to APPS can be used, demonstrating schema level privilege grants, a new Oracle AI Database 26ai feature.

S/N	Policy	User Name	Used Role	System Privileges	Object			Grant Path
					Owner	Name	Type	
1	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	DEPARTMENTS	TABLE	APPS
2	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	JOB_HISTORY	TABLE	APPS
3	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	COUNTRIES	TABLE	APPS
4	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	EMPLOYEES	TABLE	APPS
5	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	LOCATIONS	TABLE	APPS
6	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	REGIONS	TABLE	APPS
7	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	JOBS	TABLE	APPS
8	HR Analysis Policy	APPS	APPS	CREATE SESSION			(null)	APPS
9	HR Analysis Policy	APPS	PUBLIC	(null)	SYS	DBMS_APPLICATI...	PACKAGE	PUBLIC
10	HR Analysis Policy	APPS	PUBLIC	(null)	SYSTEM	PRODUCT_PRIVS	VIEW	PUBLIC
11	HR Analysis Policy	APPS	PUBLIC	(null)	SYS	DUAL	TABLE	PUBLIC

Figure 3-28: Example report on DBA_USED_PRIVS showing object access

Privilege analysis benefits

Static privilege analysis tools like DBSAT and Data Safe provide solid starting points, revealing which roles and privileges users possess. Privilege Analysis extends this by showing which privileges accounts actually use. The combination of static and dynamic analysis enables:

- **Reporting actual privilege utilization**: Understanding operational requirements based on observed behavior
- **Identifying unused privileges and roles**: Finding revocation candidates with confidence
- **Reducing risk through least privilege**: Shrinking blast radius when credentials are compromised
- **Eliminating privilege creep**: Removing accumulated permissions from role changes and temporary grants that became permanent

With deeper understanding of privileges required for applications to operate, database administrators can confidently refine granted permissions to limit unnecessary access. Reducing unnecessary privileges shrinks attack surfaces and diminishes the potential impact of stolen credentials or account misuse.

Privilege Analysis empowers organizations to transform privilege management from speculation-based guesswork to data-driven decision making. Rather than accepting privilege accumulation as inevitable, organizations can systematically reduce privileges to operational minimums, measurably reducing risk.

Having addressed privilege assessment, the final major assessment dimension is patch management: ensuring databases remain current with security updates.

Managing security patches

Patching represents a critical assessment dimension: identifying which databases lack current security updates and ensuring timely remediation. Every quarter, Oracle delivers database fixes for functional, performance, and security issues discovered through internal testing, customer reports, and external security researcher disclosures.

Security fixes address diverse vulnerability categories:

- **SQL vulnerabilities**: SQL injection weaknesses, vulnerable SQL statements, buffer overflows
- **Client vulnerabilities**: Database client software, JDBC drivers, and third-party code weaknesses
- **Cryptographic weaknesses**: Encryption algorithm vulnerabilities, weak key management, protocol weaknesses
- **Network vulnerabilities**: Network protocol weaknesses, authentication bypasses, session hijacking vectors
- **Remote code execution vulnerabilities**: Flaws enabling attackers to execute arbitrary commands

Organizations require timely patch application to maintain proper security posture. Not applying patches means knowingly accepting the risk of publicly documented vulnerabilities. Once vulnerability details become public through Oracle Critical Patch Updates or security researcher publications, attackers can rapidly develop and deploy exploits. The window between disclosure and active exploitation continues shrinking, sometimes measured in hours.

Proactive maintenance through RUs, MRPs, and CSPUs

For proactive maintenance, apply the quarterly patch bundle (Release Update / RU) available in the My Oracle Support (MOS) Customer Portal for each Oracle AI Database software release. RUs remain the foundation of Oracle AI Database patching and continue to be released quarterly, typically in January, April, July, and October.

There is also a monthly database security patch delivery vehicle called the Critical Security Patch Update (CSPU), that provides a smaller, targeted way to deliver critical security fixes between quarterly release cycles.

Use the patch vehicle appropriate for the platform:

Patch type	Primary use	Platform guidance	Cadence	Contents
Release Update (RU)	Baseline quarterly database update	All supported platforms	Quarterly	Security, regression, optimizer, and functional fixes
Monthly Recommended Patch (MRP)	Monthly cumulative patching between RUs	Linux customers should apply RUs plus MRPs	Monthly	Critical security fixes plus critical one-off fixes
Critical Security Patch Update (CSPU)	Monthly cumulative critical security patching between RUs	Non-Linux customers should apply RUs plus CSPUs	Monthly	Critical security fixes only

MRPs and CSPUs are both tied to a specific RU base. Each MRP or CSPU requires the RU it was built for and cannot be applied to a different RU stream. For example, an MRP or CSPU built for the April 2026 RU cannot be applied on top of the January 2026 RU.

Each monthly MRP or CSPU is cumulative within its RU stream. Oracle's current plan is to continue shipping six MRPs and add six CSPUs for each RU. This gives organizations a structured way to reduce exposure to critical vulnerabilities between quarterly RUs while preserving predictable maintenance cycles.

MRPs contain a superset of CSPU content. MRPs include critical security fixes and critical one-off fixes, while CSPUs include critical security fixes only. Applying either an MRP or CSPU does not change the database version number.

Oracle will provide CVE transparency for these updates by listing the CVEs addressed by each RU, MRP, and CSPU. This helps security and compliance teams map patch application to known vulnerability exposure and validate remediation status across database fleets.

- **Recommendation:** Stay current on RUs to minimize exposure to known bugs and security vulnerabilities.

Linux x86-64 guidance: If your databases run on Linux x86-64, and you have validated your application on a given RU and are ready to go live, check for the latest available MRP for that RU and apply the appropriate CSPU.

Non-Linux guidance: If your databases are not running on Linux x86-64, and you have validated your application on a given RU and are ready to go live, check for the latest available CSPUs for that RU and apply it.

Monthly Recommended Patches (MRPs)

Monthly Recommended Patches (MRPs) are provided for Oracle AI Database 26ai and Oracle Database 19c deployed on Linux x86-64 to deliver proactive patching between quarterly RUs.

Key characteristics:

- An MRP is a single downloadable merge patch for a specific RU
- It delivers a consolidated set of recommended one-off fixes, including security fixes
- Unlike an RU, an MRP does not change the release number

Coverage and content:

- Oracle publishes MRPs for an RU for the six months following that RU's release
- MRPs include:
    - All fixes documented in "Oracle Database 19c and Oracle AI Database 26ai Important Recommended One-off Patches" (MOS note KB188772)
    - All fixes from prior MRPs for the same RU (i.e., cumulative for that RU)

Why use MRPs: MRPs support a more conservative maintenance approach by aggregating critical/security/regression fixes while allowing organizations to defer moving to the next RU for up to six months, reducing security exposure and addressing critical issues in the interim.

By aggregating the latest critical and regression fixes from the prior six months, MRPs enable a more conservative maintenance strategy. Organizations can defer applying a full RU for up to six months while still reducing security exposure and addressing critical issues.

Critical Security Patch Updates (CSPUs)

Critical Security Patch Updates (CSPUs) are provided for Oracle AI Database deployments on non-Linux platforms to deliver critical security fixes between quarterly Release Updates (RUs). Beginning May 28, 2026, Oracle introduced

CSPUs as a monthly database security patch delivery vehicle for customers who need to address high-priority vulnerabilities without waiting for the next quarterly RU.

Key characteristics:

- A CSPU is a patch vehicle for a specific RU
- It delivers targeted critical security fixes in a smaller, more focused format
- Unlike an RU, applying a CSPU does not change the database version number
- CSPUs follow a numbering scheme similar to MRPs
- CSPUs are intended for non-Linux platforms; Linux customers should continue to use RUs and MRPs

Coverage and content:

- Oracle plans to release CSPUs monthly
- Each RU is expected to receive up to six CSPUs
- Each CSPU is cumulative for its specific RU stream
- CSPUs include critical security fixes only
- CSPUs do not include the broader critical one-off fixes included in MRPs
- Oracle will provide a list of CVEs addressed by each CSPU

A CSPU must be applied only on top of the specific RU it was built for. For example, a CSPU created for the April 2026 RU cannot be applied on top of the January 2026 RU.

Why use CSPUs: CSPUs give customers on non-Linux platforms a predictable monthly mechanism to address critical database security vulnerabilities between quarterly RUs. They help reduce exposure to high-priority threats while allowing organizations to stay within a controlled RU-based maintenance stream.

Table 3- compares Release Updates, MRPs and CSPUs

Criteria	Release Update (RU)	MRPs	CSPUs
Cadence	Quarterly	Monthly for long-term releases	Monthly for long-term releases
Zero downtime	RAC Rolling	RAC Rolling	
Security fixes	Included	May include CPU Alerts and fixes for vulnerabilities with high CVSS scores	Included
Regression fixes	Included	Included	Not included
Proactive functional fixes	Included	Not included	Not included
Optimizer plan changes (off by default)	Included	Not included	Not included
Minor functional enhancements	Included	Not included	Not included
Emergency one-offs	Included	Included	Not included

Supported operating systems	All supported platforms	Linux x86-64	Non-Linux

Table 3-2: Summary showing main differences between RUs, MRPs, and CSPUs

It is strongly recommended that the database and Oracle Grid Infrastructure be kept current by applying RUs to remediate known security vulnerabilities and reduce the likelihood of a successful attack. RUs bundle the latest security, regression, and critical fixes into a single update. Maintaining RU currency with MRPs/CSPUs also lowers the chance that separate interim one-off patches will be needed, which can create unique software baselines and drive ongoing, costly patch maintenance.

Oracle-managed cloud services receive security updates automatically as part of the service, but customer-managed deployments, whether on-premises or on OCI, require customers to plan, test, and apply the appropriate RU, MRP, or CSPU.

Patching tools and methodologies

Patching Oracle AI Database involves complexity that can cause downtime without proper planning and execution. Oracle recommends out-of-place patching because it supports rolling patching, granular changes, and faster rollback capabilities.

Out-of-place patching benefits include:

- **Reduced disruption through rolling patching**: Patch Oracle RAC nodes sequentially while other nodes remain operational
- **Testing before cutover**: Validate patches on standby systems before patching primary production systems
- **Rapid rollback**: Quickly revert to previous software versions if issues arise

Oracle Fleet Patching and Provisioning

Oracle Fleet Patching and Provisioning (FPP) facilitates database fleet lifecycle management through automation, standardization, and out-of-place patching methodologies. FPP drift detection verifies environment compliance, enabling early detection of unexpected changes.

FPP automates routine operations including:

- **Provisioning new clusters and databases**: Automated deployment from standardized templates
- **Installing patched Oracle binaries**: Creating new Oracle Homes with patches pre-applied
- **Patching clusters and databases**: Coordinated patching across RAC clusters
- **Performing database and cluster upgrades**: Version upgrades with minimal downtime

FPP blueprints ensure maintenance executes in correct sequences with minimal business impact. Organizations get repeatable workflows reducing human error and shortening maintenance windows.

With FPP, large organizations can patch hundreds or thousands of databases in single maintenance windows with minimal manual intervention. Smaller organizations benefit equally, leveraging identical automation to save time, enhance consistency, and reduce risk.

FPP availability depends on licensing: use FPP when database targets are licensed for Oracle Real Application Clusters (RAC) or Oracle RAC One Node. For single-instance databases, FPP is available if organizations have licensed the Enterprise Manager Database Lifecycle Management Pack.

Integrating patching with assessment

Assessment tools integrate with patching workflows to ensure comprehensive security posture management:

- **DBSAT and Data Safe** flag missing patches during security assessments
- **Security Central** tracks patch application across fleets and alerts on inconsistencies
- **DBLM** coordinates assessment with patching schedules, differentiating expected changes from unexpected drift
- **Oracle Update Advisor** gives accurate guidance on software health
- **FPP** automates patch deployment validated by assessment tools

This integration creates closed-loop processes: assessment identifies missing patches, patching tools apply updates, subsequent assessment verifies successful remediation.

Organizations building a mature security posture management program typically weave assessment, privilege analysis, and patch management into a continuous workflow: assess, prioritize, remediate, validate, repeat. This steady rhythm helps sustain consistently secure configurations, even as systems, users, and environments are in constant motion.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Database Security Assessment Tool
- Get Started with Oracle Data Safe Fundamentals
- Oracle Database Security Central workshop
- Privilege Analysis workshop
- Oracle Fleet Patching and Provisioning workshop
- Patch me if you can workshop

Summary

Database security posture management turns one-off security checks into a repeatable, defensible practice. Modern databases expose hundreds of configuration options, and a single weak setting can become an open path to data. A strong assessment program catches misconfigurations early, documents due diligence, and shows measurable improvement over time so organizations can answer tough questions from auditors, executives, and regulators with confidence.

Oracle provides a set of complementary tools to support this program at different scales and deployment models. DBSAT offers a free, consistent way to assess individual databases, identify high-risk issues, and discover sensitive data. Oracle Data Safe adds fleet-wide assessment, baselines, drift reporting, and workflows for acceptable risk in an Oracle Cloud Infrastructure service. Oracle Database Security Central delivers similar assessment and posture management capabilities for organizations that prefer or require on-premises deployment. Together, these options give security teams a flexible toolkit that fits existing environments and operational maturity.

Posture management also depends on understanding who has access and what they actually use. Privilege Analysis and related views such as DBA_USED_PRIVS and DBA_UNUSED_PRIVS reveal which privileges are truly exercised and which remain unused. That visibility helps teams enforce least privilege, reduce the blast radius of compromised credentials, and remove access that accumulated over time but no longer serves a legitimate purpose.

Keeping software current is the other pillar of a resilient posture. Delayed patching turns public vulnerability disclosures into practical attack paths. Quarterly Release Updates and, where available, Monthly Recommended Patches help structure patch cadence, while practices such as out-of-place patching, thorough pre-cutover testing, and clear rollback plans keep patch operations predictable and low risk. Many organizations combine these tools and practices into a steady operational rhythm: assess, prioritize, remediate, and validate, then repeat.

The next chapter focuses on sensitive data discovery and classification. It explains how to locate sensitive data across your database estate, categorize it accurately, and align controls, auditing, and monitoring to what that data represents for your business and regulatory obligations. With that visibility, teams can move from uncertainty to precise, risk-informed action.

Chapter 4

Sensitive Data Discovery

Introduction: The foundation of data protection

Security controls only work when teams know what they are protecting. Many organizations spend millions on encryption, firewalls, and access controls, but still struggle with a basic question: Where is our sensitive data? This chapter helps remove that uncertainty by walking through a systematic approach to discovering sensitive data in Oracle Database environments, which is the foundation for every other security control covered in this book.

In practice, discovery needs to satisfy two priorities: regulatory compliance and risk management. Regulations like GDPR, PCI DSS, HIPAA, and India's Digital Personal Data Protection Act (DPDPA) call for specific protections for different kinds of data, but those requirements do not mean much if teams cannot first find the data in scope. On the risk side, the likely impact of a breach should drive how strong the controls need to be. Discovery makes that possible by helping organizations focus strict safeguards where the risk is highest, while avoiding over-investing in areas where the data is lower risk.

The chapter first explains why sensitive data matters for business and security outcomes, then highlights why AI applications increase the urgency. It defines sensitive data, outlines the methods used to locate it, and then provides detailed guidance on Oracle's four discovery tools: Database Security Assessment Tool (DBSAT), Oracle Data Safe, Oracle Database Security Central (Security Central), and Enterprise Manager's Application Data Modeling.

Why sensitive data matters

Databases are where modern enterprises keep the information that keeps everything moving: customer transactions, product designs, financial records, research findings, and operational metrics. That data enables organizations to serve customers well, make smarter decisions, reduce costs, and preserve competitive advantage. The challenge is that much of it is also sensitive or personal, such as customer records, payment details, healthcare information, employee compensation, and proprietary business metrics.

The business value and business risk of data

Because sensitive data is valuable to the business, it is also valuable to threat actors. As sensitivity increases, so does its appeal for theft, misuse, or ransom. A breach involving customer payment information can hit both revenue and reputation. Exposed intellectual property can chip away at competitive positioning. Compromised healthcare records can lead to regulatory penalties and lawsuits. The equation is simple: high-value data attracts high-intent adversaries and results in high-impact consequences when exposed.

That is why sensitive data protection needs to be a practical priority, not just a policy statement. When security teams know what is sensitive and where it lives, they can apply stronger controls exactly where they will matter most. This focused approach helps protect customers, safeguard intellectual property, meet compliance obligations, and keep the business running.

Discovery as the first step in defense

The security axiom holds true: organizations cannot protect what they cannot see. The time, budget, and effort invested in database security should reflect both regulatory obligations and business risk exposure. The higher the potential impact of a compromise, the stronger the controls required to reduce risk to acceptable levels.

Discovery is typically the foundation for most subsequent security measures. Before implementing encryption, masking, access controls, or audit policies, security teams must answer three foundational questions:

- What types of sensitive data exist in the environment?
- How much sensitive data is present across all databases?
- Where exactly does this data reside—which databases, schemas, tables, and columns?

Without precise answers to these questions, security efforts become inefficient at best and ineffective at worst. Organizations may over-invest in protecting low-risk data while leaving critical information exposed. They may fail audits because they cannot demonstrate where regulated data resides. They may deploy masking solutions that break application dependencies because referential relationships were not mapped. Discovery eliminates these blind spots by providing the visibility required for risk-based security decisions.

The AI imperative: New urgency for discovery

The rapid adoption of artificial intelligence capabilities has fundamentally changed the risk landscape for sensitive data. Organizations are deploying AI-powered applications, implementing Retrieval-Augmented Generation (RAG) systems, and connecting large language models directly to production databases to deliver intelligent user experiences. While these technologies create substantial business value, they also introduce new pathways for unintended data exposure.

AI applications and the exposure surface

RAG systems retrieve relevant information from databases and pass it to AI models as context for generating answers. When access controls are weak or data classification is incomplete, RAG systems can expose sensitive information to unauthorized users. For example, an employee might ask a routine HR benefits question and receive executive compensation details because those records are stored in the same HR data source. Without data discovery and classification, the AI application has no reliable way to distinguish information that can be shared from information that must remain confidential. That gap turns routine questions into accidental disclosure paths.

The same concern shows up during AI fine-tuning and training on enterprise data. If the training dataset contains personally identifiable information (PII) or protected health information (PHI), that regulated data can end up embedded in the model weights. After that, it may be possible to pull the information back out through prompt injection attacks, model inversion techniques, or even straightforward inference from model outputs. Discovery reduces this risk by flagging sensitive data before it enters the training pipeline, so organizations can exclude it, anonymize it, or generate synthetic data as needed.

Model Context Protocol and database integration

The introduction of the Model Context Protocol (MCP) has accelerated the integration timeline between AI applications and enterprise databases. MCP provides a standardized framework for connecting AI models, such as Claude, GPT-4, or custom enterprise models, directly to databases and other enterprise resources through uniform APIs. While this simplifies integration and enables rapid development of AI-powered applications, it also means databases can be exposed to AI systems faster than security teams can implement appropriate controls.

When organizations deploy an MCP server that connects an AI model to their database, discovery becomes the first line of defense. Security teams must identify exactly what data the AI can access before users begin querying it. Discovery results inform several critical decisions:

- Which schemas and tables should be excluded entirely from the MCP context
- Where to implement row-level security policies that MCP queries must respect
- Which columns require masking or tokenization within AI responses
- What fine-grained access controls should apply based on user identity and role

The MCP server must use a restricted service account following the principle of least-privilege. This proactive approach allows organizations to provide AI systems with the data access they need for useful functionality while preventing them from retrieving information they should never see. Discovery transforms AI integration from a potential security liability into a calculated, controlled capability.

The same discovery methodologies used for traditional database security-scanning column names, analyzing data patterns, mapping referential relationships-apply equally to preparing data for AI use cases. This consistency allows security teams to leverage existing discovery investments while addressing emerging AI-related risks.

Defining sensitive data

Sensitive data is any information that, if exposed or misused, could harm individuals, organizations, or both. A practical test for sensitivity is: Would exposing this data cause reputational damage, regulatory penalties, financial loss, or personal harm if published openly? If the answer is yes, the data qualifies as sensitive and requires protection from unauthorized access whether by external attackers or internal users exceeding their authorization.

Organizations typically classify sensitive data into a few broad categories, based on the type of information involved and the regulations that apply. Figure 4-1 highlights common categories and examples of representative data types within each category.

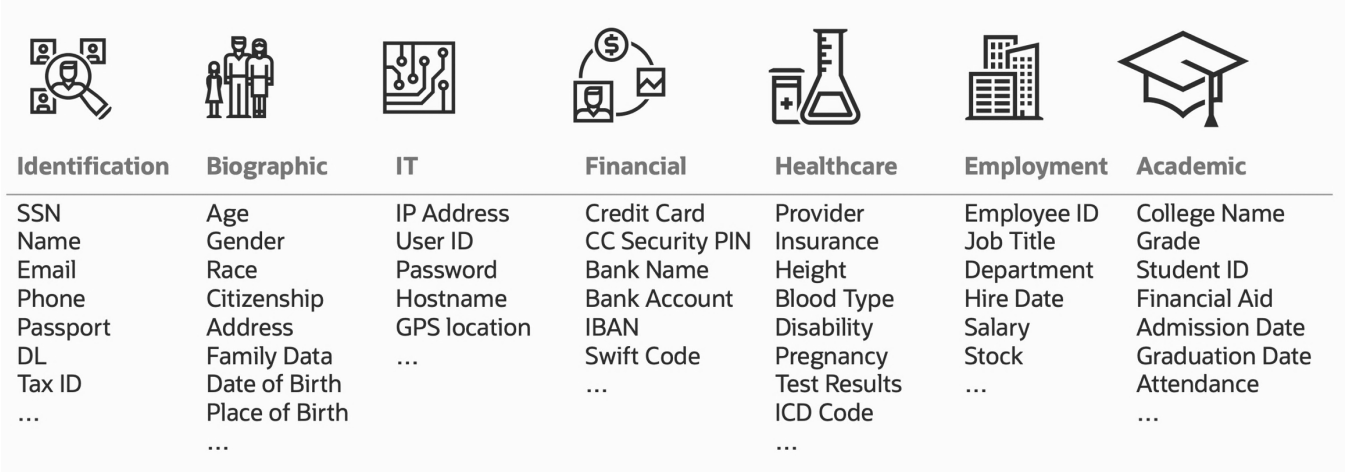


Figure 4-1: Common sensitive data categories and representative types

Organizations rarely deal with only one category of sensitive data. Different regulations spotlight different data types. For example, PCI DSS prioritizes payment account data, Sarbanes Oxley focuses on financial records, and GDPR emphasizes identification and biographic data. Understanding which categories apply in each environment helps organizations focus controls where they matter most.

Discovery methodologies: How to find sensitive data

Oracle's discovery tools employ two complementary approaches to locate sensitive data within databases. Most effective discovery strategies combine both methodologies to maximize coverage and accuracy.

Metadata-based discovery

Metadata-based discovery reviews the structural details of database objects, specifically column names and column comments, rather than the actual data values stored in tables. It looks for keywords, patterns, and naming conventions that may indicate sensitive information.

How it works: The discovery tool queries the data dictionary to retrieve column names and associated comments across all accessible schemas. It then applies pattern matching using regular expressions or keyword lists. For example, column names containing `SSN`, `CREDIT_CARD`, `SALARY`, `DOB` (date of birth), or `PASSWD` triggers a match. Multilingual keyword libraries extend coverage to databases using non-English naming conventions.

Advantages: Metadata scanning is extremely fast and lightweight because it reads only data dictionary tables rather than potentially billions of rows within user tables. It consumes minimal system resources and can be executed

against production databases with negligible performance impact. The approach works well in environments where developers follow consistent naming conventions for sensitive columns.

Limitations: Metadata-based discovery depends entirely on meaningful naming. If developers use generic or cryptic column names such as `FIELD_17`, `ATTRIBUTE_A`, or `DATA_VALUE` to store credit card numbers or Social Security numbers, metadata scanning will miss that sensitive data entirely. Additionally, this approach generates false positives when column names suggest sensitivity, but the actual content is innocuous (for example, a column named `CUSTOMER_NAME` that stores product category names).

Pattern-based discovery (data sampling)

Pattern-based discovery analyzes the actual values stored within database columns by sampling row data and applying pattern recognition algorithms. This approach identifies sensitive data based on content rather than metadata.

How it works: The discovery tool executes queries to sample a subset of rows from each table (typically a configurable percentage or fixed count). It then examines the retrieved values using various detection techniques:

- Regular expressions to match formats like credit card numbers (16 digits with optional separators), email addresses, phone numbers, or Social Security numbers
- Algorithmic validations such as checksum validation for national ID formats or credit card numbers
- Statistical analysis to detect patterns like consistent length or structure across sampled values
- Predefined templates for country-specific identifiers (e.g., Canadian Social Insurance Numbers, UK National Insurance Numbers, Indian Aadhaar numbers)

Advantages: Pattern-based discovery finds sensitive data regardless of how columns are named, making it effective in environments with inconsistent or poor naming conventions. It provides higher confidence results because detection is based on actual content rather than inferred meaning. The approach also discovers sensitive data that was added to generic columns after database design (for example, when a `NOTES` text field later begins storing customer phone numbers).

Limitations: Data sampling requires more compute resources and time than metadata scanning because the tool reads and analyzes table rows. In very large, multi-terabyte databases, sampling all tables may take hours or days. Organizations should also consider data residency requirements. If regulatory or internal policies prohibit copying sensitive values outside the database, sampling results must be protected appropriately or disabled.

Combined methodology for optimal results

The most effective discovery strategy is to combine both approaches. When a column name suggests sensitivity and pattern analysis confirms that content matches expected formats, confidence in the detection increases substantially. For example, a column named `CREDIT_CARD_NUMBER` that also contains 16-digit values passing algorithmic validation is almost certainly storing payment card data.

Conversely, when metadata and content analysis disagree, human review can adjudicate. A column named `ACCOUNT_NUMBER` containing random alphanumeric values may be a product `SKU` rather than a bank account. A column named `DATA_1` containing 9-digit values matching Social Security number patterns may require investigation to determine whether those values are actual SSNs or other identifiers.

All four Oracle discovery tools discussed in this chapter employ one or both of these methodologies, with varying levels of sophistication and configurability.

Choosing the right solution

Oracle provides four complementary tools for discovering sensitive data in Oracle Database environments, each with distinct capabilities, deployment models, and ideal use cases. Table 4-2 summarizes the four tools and their discovery

method. Organizations often combine these tools, for example, using DBSAT for a fast initial assessment and then adopting Data Safe for ongoing monitoring and management.

Tool	Primary Use Cases	Discovery Methodology
DBSAT	Quick assessments, audit preparation, lightweight reporting	Metadata (column names and comments)
Oracle Data Safe	Comprehensive discovery, continuous monitoring, cloud-native management	Metadata + pattern-based data sampling
Enterprise Manager Data Masking and Subsetting	EM Pack to protect sensitive data by masking and reduce storage requirements by subsetting data	Metadata + pattern-based data sampling
Oracle Database Security Central	Security posture management, activity monitoring, compliance reporting, firewall policy configuration	Metadata (column names and comments)

Table 4-2: Oracle sensitive data discovery tools support different use cases

The following sections examine each tool in detail, providing technical architecture, capabilities, use cases, and practical guidance for implementation.

Discovering sensitive data using DBSAT

Database Security Assessment Tool (DBSAT) is a command-line utility designed for rapid security posture assessment and sensitive data discovery. As introduced earlier, DBSAT provides comprehensive assessment by combining configuration analysis, privilege analysis, and sensitive data discovery in a single lightweight tool.

Overview and architecture

DBSAT operates as a standalone utility that connects to a target database using standard Oracle Net protocols. It requires minimal privileges, typically a read only account with access to data dictionary views, and can be executed from any system with network connectivity to the database. The tool gathers metadata on database configuration, user privileges, and sensitive column names, then processes the information locally to produce comprehensive reports.

Discovery capabilities

DBSAT identifies sensitive data by scanning column names and comments against an extensible library of sensitive data definitions. It ships with prebuilt patterns for common sensitive types in English, and additional libraries are available for seven major European languages: Dutch, French, German, Greek, Italian, Portuguese, and Spanish. In multilingual environments, organizations can deploy multiple language libraries at the same time.

Rather than sampling actual data values, DBSAT estimates the quantity of sensitive data by reading table statistics from the data dictionary, specifically the row counts maintained by the database optimizer. This method delivers volume estimates without requiring access to the underlying table data, which makes DBSAT a practical fit for environments with strict data residency requirements.

DBSAT groups discovered sensitive columns into predefined categories such as Biographic Information, Financial Information, Health Information, Identification Information, IT Information, and Job Information. Each category includes subcategories to support more precise classification. For instance, the Financial Information category includes subcategories for Bank Data, Card Data, and Tax Data.

Report formats and content

DBSAT generates detailed sensitive data assessment reports in both HTML and spreadsheet formats, making results accessible to both technical and business audiences. The reports include several key sections:

Summary statistics: High-level counts of sensitive tables, columns, and estimated rows, organized by category and subcategory. This overview helps stakeholders quickly understand the scale and distribution of sensitive data across the database. Table 4-3 shows a sample summary section.

Sensitive Category	# Sensitive tables	# Sensitive columns	# Sensitive rows
BIOGRAPHIC INFO - ADDRESS	9	36	6307209
BIOGRAPHIC INFO – EXTENDED PII	2	2	2000
FINANCIAL INFO – BANK DATA	2	2	830
FINANCIAL INFO – CARD DATA	7	7	3235
HEALTH INFO – PROVIDER DATA	1	1	149
IDENTIFICATION INFO – NATIONAL IDS	2	6	2000
IDENTIFICATION INFO – PERSONAL IDS	3	3	405
IT INFO – USER DATA	13	15	13228
JOB INFO – COMPENSATION DATA	10	12	3380
JOB INFO – EMPLOYEE DATA	8	16	406
JOB INFO – ORG DATA	5	6	278
Total	**29**	**132**	**8617644**

Table 4-3: Example DBSAT sensitive data summary by category

Each sensitive category has a preconfigured risk level (high, medium, or low), and this risk rating can be customized if desired. The report then recommends how to protect sensitive data based on its assigned risk level.

Figure 4-2 illustrates several recommendations for safeguarding high risk data, such as personal health information.

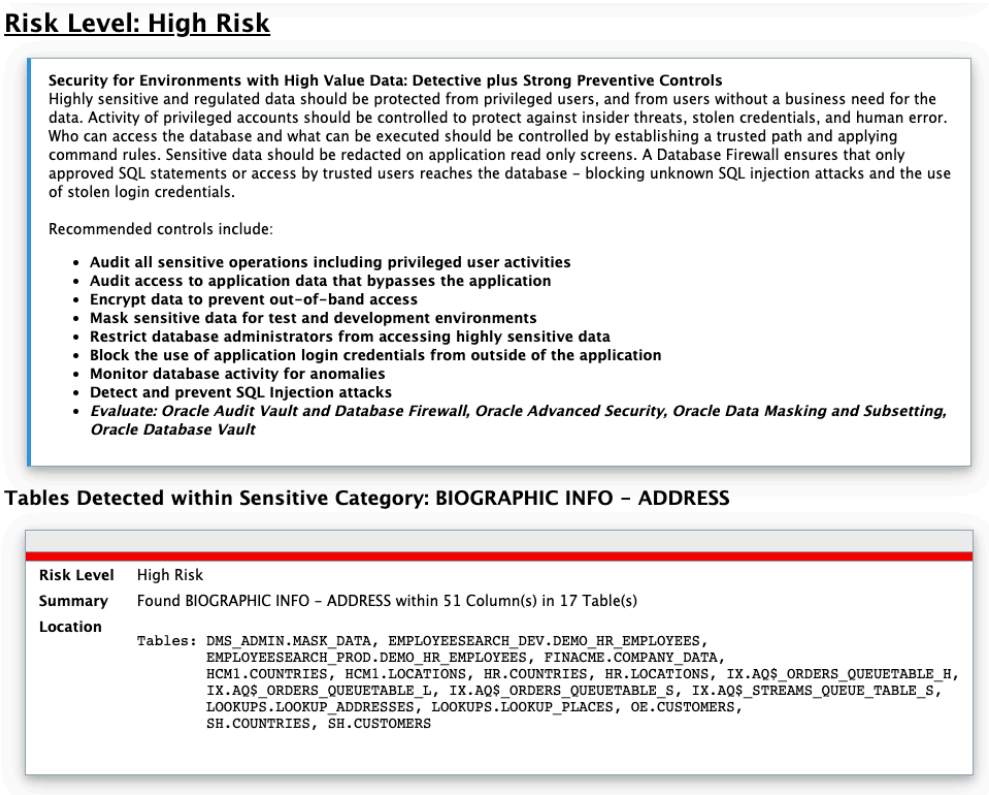


Figure 4-2: Example recommendations for protecting sensitive data

Risk-based recommendations: Each sensitive category has a preconfigured risk level that drives recommended protections. For high-risk data such as health information or payment card data, DBSAT recommends implementing encryption, restricting access through Database Vault realms, enabling audit policies, and considering data masking for nonproduction environments. Medium-risk data recommendations typically include access controls and auditing. Low-risk data may require only basic access management.

Detailed findings: Schema-level, table-level, and column-level details provide the specific locations of discovered sensitive data. This granular information enables security teams to implement targeted controls.

Use cases and deployment scenarios

DBSAT excels in several scenarios:

- **Rapid initial assessment:** When organizations need a quick inventory of sensitive data without deploying complex infrastructure, DBSAT provides results within minutes to hours depending on database size.
- **Audit preparation:** Before regulatory audits, DBSAT reports demonstrate where sensitive data resides and what controls are in place, helping organizations prepare documentation and address gaps.
- **Pre-migration analysis:** Before migrating databases to cloud environments or upgrading to new versions, DBSAT helps teams understand what sensitive data will be affected and plan appropriate protections.
- **Air-gapped environments:** Because DBSAT operates as a standalone utility without external dependencies, it works well in isolated or high-security environments where cloud-based tools cannot be deployed.

Discovering sensitive data using Data Safe

Oracle Data Safe is a comprehensive, cloud-native database security service that provides unified management for security assessment, user assessment, sensitive data discovery, data masking, activity auditing, and security monitoring. Data Safe operates as a managed service within Oracle Cloud Infrastructure but can monitor and protect databases across multiple environments including Oracle Cloud Infrastructure, third-party clouds (AWS, Azure, Google Cloud), and on-premises.

Overview and architecture

Data Safe's architecture separates the control plane (management console and services running in Oracle Cloud) from the data plane (target databases being monitored). Communication between Data Safe and target databases occurs through secure, outbound-initiated connections, ensuring that no inbound network paths need to be opened to protected databases. This design accommodates stringent network security requirements common in enterprise environments.

Discovery capabilities

Data Safe provides the most comprehensive discovery capabilities among Oracle's tools, combining metadata scanning with sophisticated pattern-based analysis of actual data values. The service includes pre-built sensitive type definitions for over 150 data types, including country-specific identifiers for Brazil, Canada, France, Germany, India, Italy, México, Netherlands, Portugal, Spain, the United Kingdom, and the United States.

Organizations can also define custom sensitive types that combine:

- Column name patterns (regular expressions)
- Column comment patterns (regular expressions)
- Data value patterns (regular expressions)
- Row sample size for validation
- Minimum match percentage thresholds

For example, a financial institution might create a custom sensitive type for internal account numbers that follow a proprietary format, specifying both the expected column naming convention and the data pattern to validate against actual values.

When Data Safe executes a discovery job, it scans column names and comments first to identify candidates, then optionally samples actual data values to confirm matches. This two-stage approach balances thoroughness with efficiency. Organizations concerned about copying sensitive values outside the database can disable data sampling entirely and rely solely on metadata scanning.

Figure 4-3 shows a sample data discovery report generated by Data Safe.

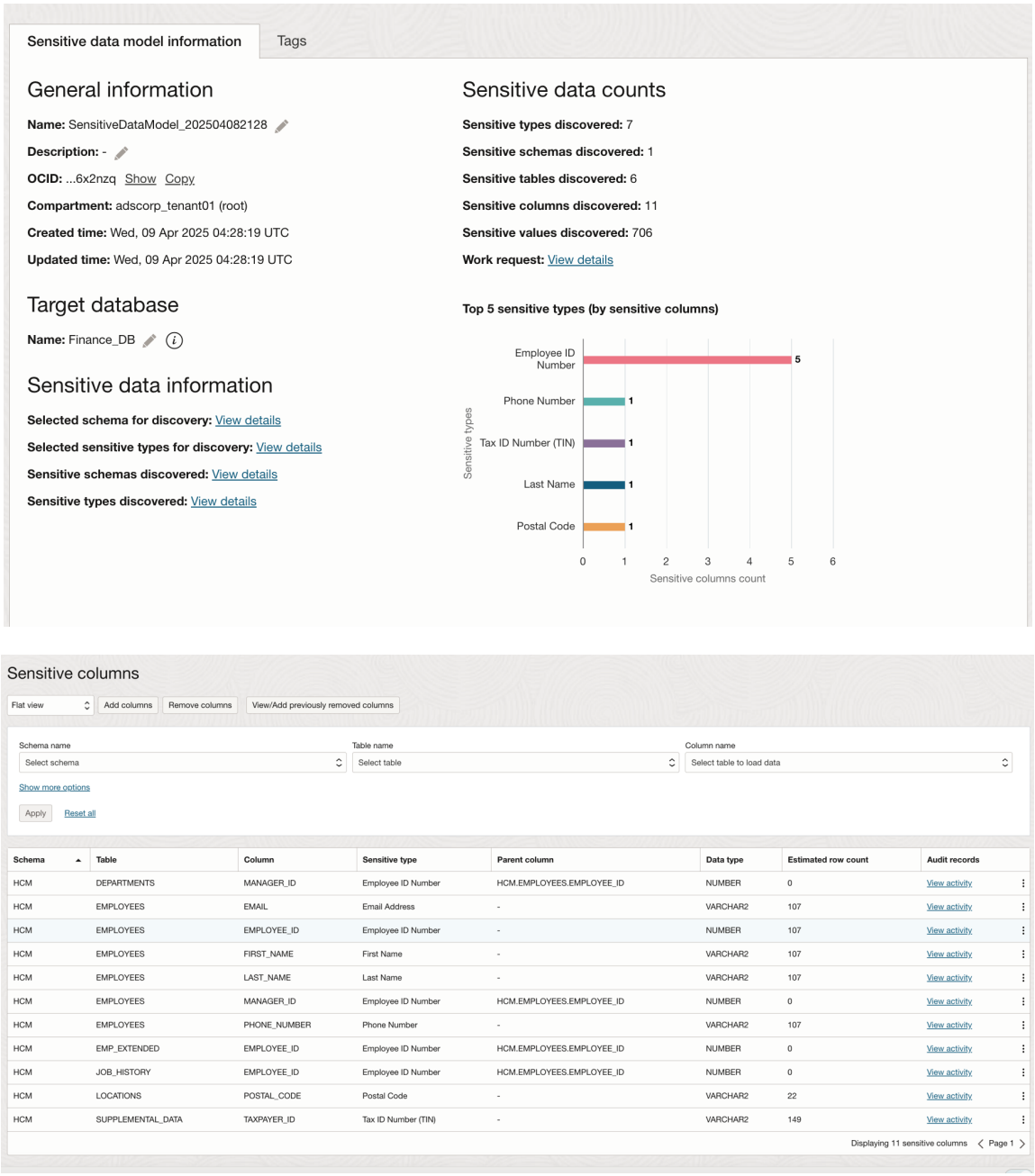


Figure 4-3: Example data discovery report in Oracle Data Safe

Data Safe scans column names, comments, and data values to help users quickly spot sensitive attributes and understand their context. It also shows how tables relate to each other by identifying relationships between primary and foreign key columns.

In addition to what is stored in the data dictionary, Data Safe helps uncover referential relationships defined at the application layer. This provides a more complete view of how data moves through the environment, enabling teams to plan masking and testing strategies without breaking dependencies.

From the Data Safe console, users can monitor activity and drill down into audit records for the data being assessed (refer to rightmost column of Figure 4-3). That makes it easier to investigate who accessed what, when, and how.

Sensitive data models in Data Safe

Securing sensitive data begins with understanding what exists and how it connects. Data Safe provides that visibility by transforming discovery results into a reusable sensitive data model that can be refined and improved over time.

Data Discovery reports summarize sensitive tables, columns, and values, and they provide detailed information for each sensitive column. Data Safe categorizes these columns by sensitive type and combines that classification with primary and foreign key relationships to build a sensitive data model. That model is revisited later in the book during the discussion on data masking, where those relationships help ensure masking is applied consistently without breaking dependencies.

The model can also store optional metadata, such as sample data and estimated row counts, to help teams understand the scale of each sensitive type across target databases, as shown in Figure 4-3. Because many organizations prefer not to copy actual values outside the database, Data Safe defaults to avoiding the collection of sample data. For a clear view of exactly what Data Safe collects from databases, refer to the reference section of the Data Safe Administrator's guide.

As the target database evolves, incremental updates can be used to keep the sensitive data model up to date. This approach helps refine the model over time by adding or removing sensitive columns and maintaining the referential relationships between those columns. Discovery can also begin with a narrower scope, such as scanning a single sensitive category or a set of selected schemas, and then gradually expand through incremental discovery to broaden coverage as the model matures.

For portability, a sensitive data model can be downloaded, updated offline, and uploaded to the same or a different Oracle Data Safe region. Each sensitive data model is associated with one target database at a time, and that association can be changed whenever needed.

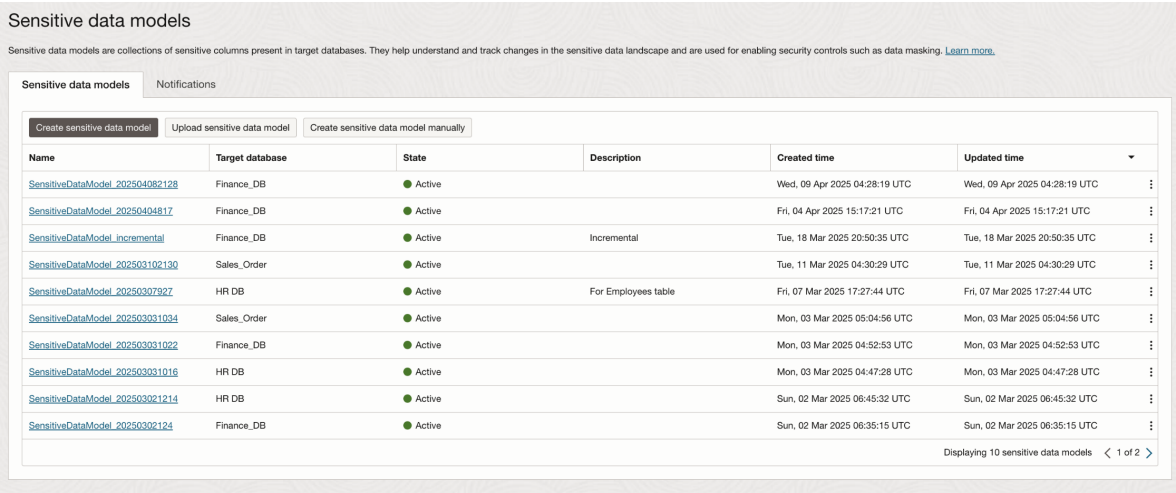


Figure 4-4: Example data discovery sensitive data models

Sensitive data models help inform the design of other security controls, such as data masking, auditing, Database Vault policies, and fine-grained access controls. For example, a masking policy can be defined using a sensitive data model to mask sensitive data in nonproduction database clones, such as test and development instances.

Use cases and deployment scenarios

Oracle Data Safe is the recommended solution for organizations seeking comprehensive, ongoing discovery and management of sensitive data, particularly in the following scenarios:

- **Cloud-native databases:** For Oracle Autonomous Database, Oracle Database Cloud Service, or Oracle Exadata Cloud Service deployments, Data Safe provides native integration without requiring additional infrastructure.
- **Hybrid and multi-cloud:** Organizations with databases distributed across Oracle Cloud, AWS, Azure, Google Cloud, and on-premises can manage discovery centrally through a single Data Safe instance.
- **Continuous monitoring:** When organizations require ongoing visibility into sensitive data as databases evolve, Data Safe's scheduled discovery jobs and incremental updates maintain current models.
- **Masking workflows:** Organizations that mask sensitive data in nonproduction environments benefit from Data Safe's tight integration between discovery models and masking policies.
- **Compliance reporting:** When auditors require evidence of where regulated data resides and how it is protected, Data Safe provides comprehensive reports with drill-down capabilities to supporting evidence.
- **API-Driven automation:** Organizations implementing DevSecOps practices can use Data Safe's REST APIs to automate discovery workflows, integrate results into security information systems, and trigger downstream security controls.

Data residency and privacy considerations

Because Data Safe is a cloud service, organizations sometimes raise concerns about what data is collected and where it is stored. By default, Data Safe collects only metadata about sensitive columns like names, types, categories, and estimated counts, not actual sensitive values. Organizations can optionally enable sample data collection for validation purposes, but this feature is disabled by default.

All information collected by Data Safe is stored within the Oracle Cloud Infrastructure region where the Data Safe instance is provisioned. Organizations subject to data residency requirements can select appropriate regions during setup. For detailed information about what data that Data Safe collects and stores, Oracle provides comprehensive documentation in the Data Safe Administrator's Guide.

Discovering sensitive data using Oracle Database Security Central

Oracle Database Security Central (Security Central) is an integrated solution that combines comprehensive security posture management, audit data collection, centralized audit analysis, and SQL firewall capabilities. While Security Central's primary purpose is monitoring database activity and enforcing security policies, it also includes sensitive data discovery functionality that enhances its monitoring and reporting capabilities.

Overview and architecture

Security Central consists of two main components:

- **Audit Vault Server:** A hardened Linux appliance (physical or virtual) that collects audit data from multiple database sources, normalizes the data into a central repository, and provides reporting and analysis tools.
- **Database Firewall:** A network-inline or monitor-mode appliance that inspects SQL traffic between applications and databases, enforcing policy-based controls and generating additional audit events.

Discovery capabilities

Security Central's discovery functionality focuses on metadata-based scanning of column names and comments, like DBSAT. When a target is registered in the Security Central console, a sensitive data discovery job runs automatically on first registration. You can schedule it to run periodically. The job queries the data dictionary of registered target databases to identify columns matching predefined or custom sensitive type patterns.

Unlike Data Safe and Enterprise Manager, Security Central does not sample actual data values for pattern matching. This design decision reflects Security Central's primary use case: enabling the monitoring and policy systems to recognize when sensitive data is being accessed, without requiring Security Central itself to access that sensitive data.

Discovered sensitive objects populate Security Central's database inventory, where they are tagged with sensitive types and categories. This metadata enriches several capabilities including:

- Reports can filter and highlight activity on sensitive data
- Policies can reference sensitive data
- Compliance reports can demonstrate coverage of sensitive data access monitoring
- Alerts can trigger when specific sensitive data is accessed by privileged users or during non-business hours

Data privacy reports

Security Central provides specialized reports that leverage discovered sensitive objects to answer compliance and security questions:

Sensitive data inventory: Lists all discovered sensitive objects across monitored databases, organized by target, schema, table, column, and sensitive type. This report demonstrates to auditors that the organization knows where sensitive data resides.

Activity on sensitive data: Shows detailed access patterns for sensitive objects, including user identity, session details, SQL statements executed, and timestamps. This report supports investigations and compliance requirements to demonstrate that sensitive data access is monitored and appropriate.

Activity on sensitive data by privileged users: A filtered view focusing specifically on access by users with elevated privileges (DBAs, application admins, security officers). Because privileged users often have broad access rights, monitoring their activity on sensitive data provides an additional layer of oversight to detect potential misuse.

These reports support compliance requirements for regulations such as GDPR (demonstrating data access monitoring), PCI DSS (logging access to cardholder data), HIPAA (tracking access to protected health information), and SOX (monitoring access to financial data).

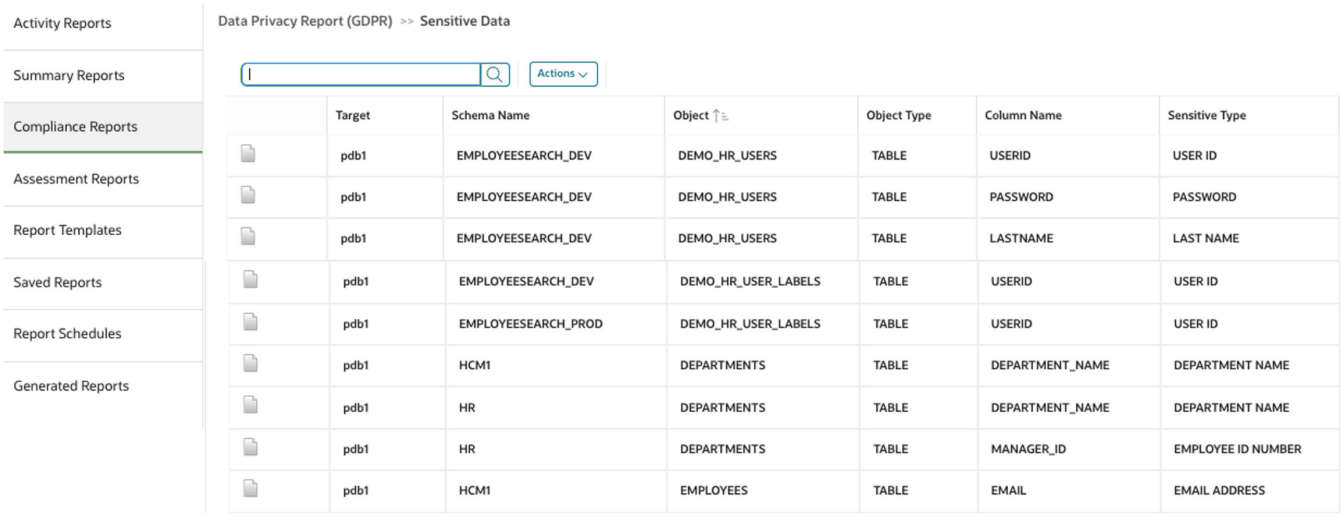


Figure 4-5: Example Security Central sensitive data report

Integration with policies

Security Central can use discovered sensitive objects as criteria in Database Firewall and audit policies. For example, an organization can create policies that:

- Block SQL statements that select from tables containing payment card data, unless they originate from approved application IP addresses
- Generate alerts when administrators query tables containing customer personal information outside approved business hours
- Trigger alerts when a query returns more rows of sensitive data than a predefined threshold
- Monitor data manipulation language (DML) activity on sensitive data

This integration transforms discovery from a purely informational exercise into an active component of the security enforcement architecture.

Use cases and deployment scenarios

Security Central's discovery capabilities serve organizations that prioritize activity monitoring and real-time policy enforcement:

- **Compliance programs:** Organizations subject to audit requirements can use sensitive data reports to demonstrate comprehensive monitoring of regulated data access.
- **Privileged user oversight:** When concerns exist about insider threats or privileged user activity, Security Central tracks access to sensitive data by administrators and generates alerts for suspicious patterns.
- **Network-layer protection:** In environments where application-layer or database-layer controls are insufficient, Database Firewall provides an additional enforcement point that can block or mask sensitive data based on discovery results.
- **Centralized audit:** Organizations with heterogeneous database environments (Oracle, SQL Server, PostgreSQL, MySQL, MongoDB) can use Security Central to collect and correlate audit data centrally, with sensitive object discovery providing context across all platforms.

Discovering sensitive data using Enterprise Manager

Oracle Enterprise Manager is a comprehensive systems management platform that provides monitoring, administration, and lifecycle management for Oracle environments. With Oracle Enterprise Manager, the data discovery module can be used to quickly identify sensitive columns, making it easier to prioritize the right controls. The **Data Masking and Subsetting** pack within Enterprise Manager includes a data discovery module that identifies sensitive columns and models application data relationships.

Overview and architecture

Enterprise Manager's discovery functionality is particularly relevant for organizations that:

- Operate primarily on premises with requirements to keep all security data within their own data centers
- Already use Enterprise Manager for database administration and operations management
- Need tight integration between discovery, masking, and other database lifecycle operations
- Manage complex application environments where understanding data relationships is critical

Data discovery, especially the sensitive types feature, is part of Oracle Data Masking and Subsetting, an Enterprise Manager pack. Licensing for data discovery is also included with database security options such as Advanced Security, Database Vault, and Label Security. Like Oracle Data Safe, Enterprise Manager examines column names, comments, and data values to discover sensitive columns.

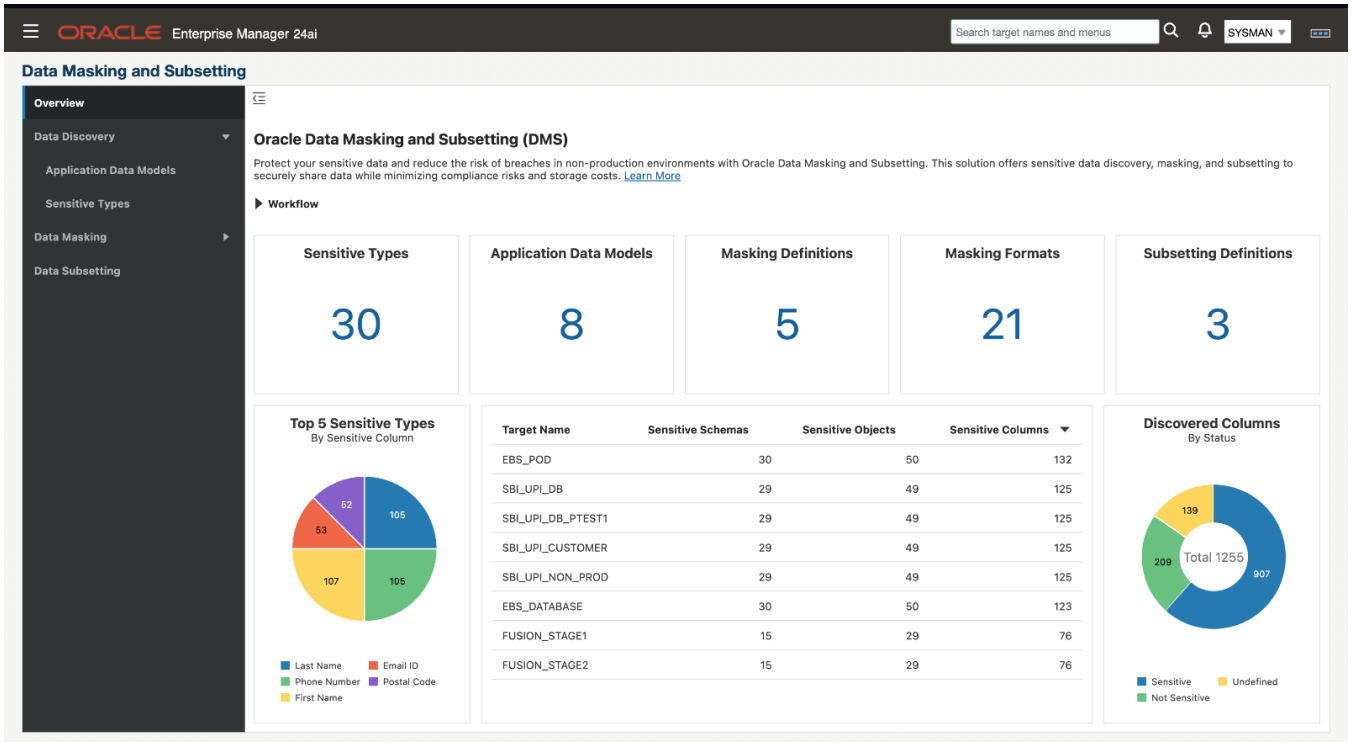


Figure 4-6: Example sensitive data discovery in Enterprise Manager

Referential relationship discovery

Data discovery uses its sensitive types feature to perform pattern matching and identify sensitive columns. Schedule a sensitive data discovery job to scan the database for data that might need additional protection. One of Enterprise Manager's distinguishing capabilities is its sophisticated analysis of referential relationships between database objects. The tool automatically discovers relationships defined through foreign key constraints in the database schema. Additionally, it can detect application-level relationships that are not explicitly defined in the database but exist in application logic.

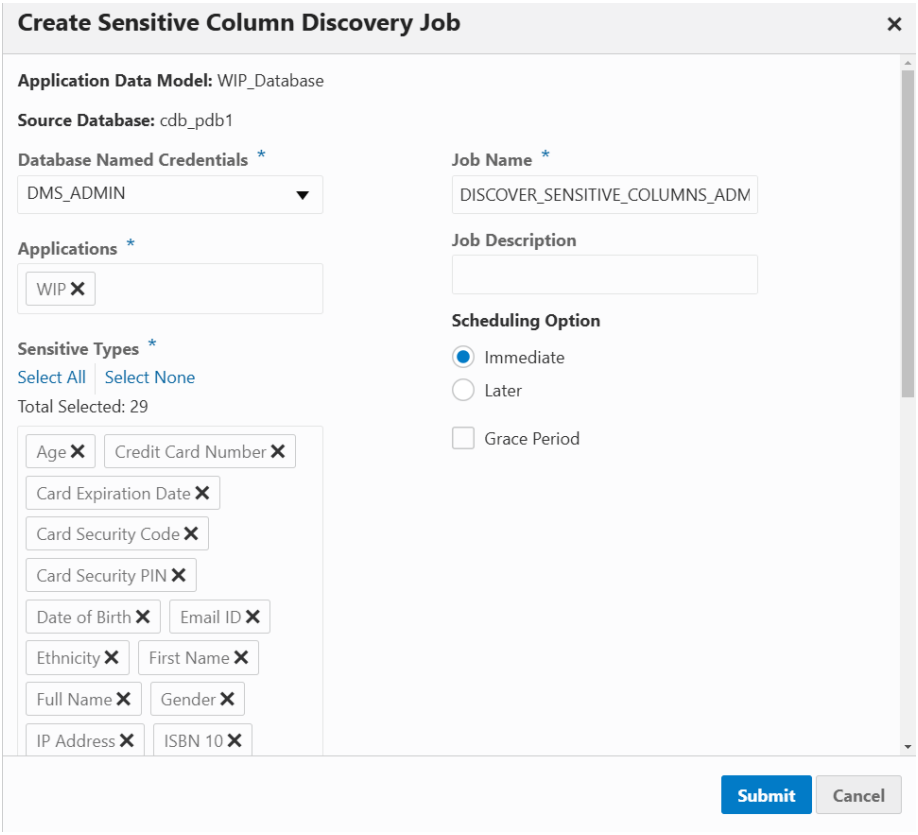


Figure 4-7: Example screen illustrating the scheduling of a sensitive column discovery job

Application-level relationship discovery works by analyzing data patterns: if a column in one table contains values that consistently match values in another table's column, the tool infers a likely relationship. Administrators can review these inferred relationships and confirm them for inclusion in the Application Data Model.

Application data models

The discovery process culminates in an Application Data Model (ADM)—a comprehensive representation of sensitive data and its relationships within a specific application context. The ADM contains:

- All discovered sensitive columns with assigned types and categories
- Database-defined referential relationships (foreign keys)
- Application-level referential relationships (inferred or manually defined)
- Parent-child hierarchies showing data dependencies
- Metadata about discovery job execution and last update times

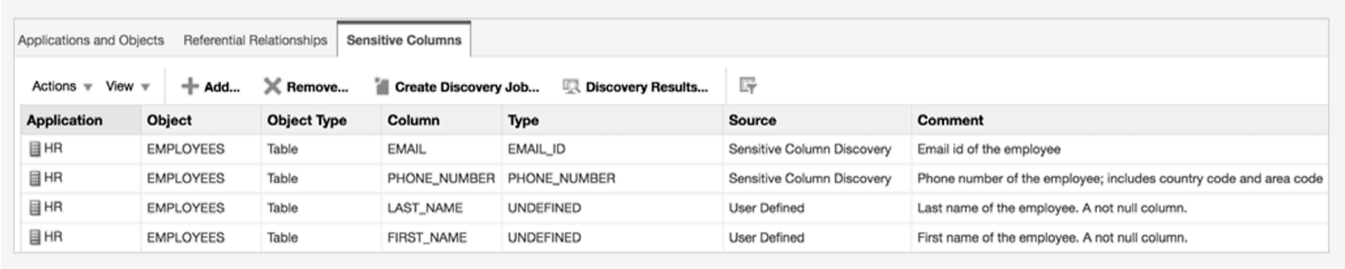
Application Data Models are stored in the Enterprise Manager repository and can be reused across multiple security operations:

Data Masking: The ADM serves as input to Enterprise Manager's data masking engine, which uses relationship information to maintain consistency when masking related columns across tables.

Database Vault policies: Security teams can use ADM information to define Database Vault realms that protect sensitive schemas or tables, command rules that restrict operations on sensitive data, and factors that provide context-aware authorization.

Test Data Management: When creating nonproduction database clones, organizations can use ADMs to identify which data must be masked or subsetted to reduce storage requirements while maintaining application functionality.

The discovered sensitive columns can be reviewed and, when needed, additional columns can be manually added to the list. Figure 4-8 shows a list of discovered as well as user-defined sensitive columns.



Application	Object	Object Type	Column	Type	Source	Comment
HR	EMPLOYEES	Table	EMAIL	EMAIL_ID	Sensitive Column Discovery	Email id of the employee
HR	EMPLOYEES	Table	PHONE_NUMBER	PHONE_NUMBER	Sensitive Column Discovery	Phone number of the employee; includes country code and area code
HR	EMPLOYEES	Table	LAST_NAME	UNDEFINED	User Defined	Last name of the employee. A not null column.
HR	EMPLOYEES	Table	FIRST_NAME	UNDEFINED	User Defined	First name of the employee. A not null column.

Figure 4-8: Example showing sensitive columns discovered using Enterprise Manager ADM

Data discovery analyzes the referential relationships between application objects by using foreign key constraints defined in the database. It can also automatically discover application level referential relationships that are not defined in the database. The discovered referential relationships can be reviewed, and additional relationships can be added manually when needed.

Understanding these dependencies helps protect application integrity during data masking by ensuring data in related columns is masked consistently. Figure 4-9 shows a list of parent-child relationships found inside the data dictionary.

Application	Object	Columns	Key Type	Source
HR				Dictionary
HR	COUNTRIES	COUNTRY_ID	Parent	Dictionary
HR	LOCATIONS	COUNTRY_ID	Dependent	Dictionary
HR	DEPARTMENTS	DEPARTMENT_ID	Parent	Dictionary
HR	EMPLOYEES	DEPARTMENT_ID	Dependent	Dictionary
HR	JOB_HISTORY	DEPARTMENT_ID	Dependent	Dictionary
HR	EMPLOYEES	EMPLOYEE_ID	Parent	Dictionary
HR	DEPARTMENTS	MANAGER_ID	Dependent	Dictionary
HR	EMPLOYEES	MANAGER_ID	Dependent	Dictionary
HR	JOB_HISTORY	EMPLOYEE_ID	Dependent	Dictionary
OE	CUSTOMERS	ACCOUNT_MGR_ID	Dependent	Dictionary
OE	ORDERS	SALES_REP_ID	Dependent	Dictionary

Figure 4-9: Example showing referential relationships

Figure 4-10 puts everything together. Data discovery automatically discovers sensitive columns, database-defined referential relationships, and application-level referential relationships and creates an application data model, which it then stores in the Enterprise Manager repository. This application data model can be used to help implement security controls such as Oracle Data Masking and Subsetting or Oracle Database Vault.

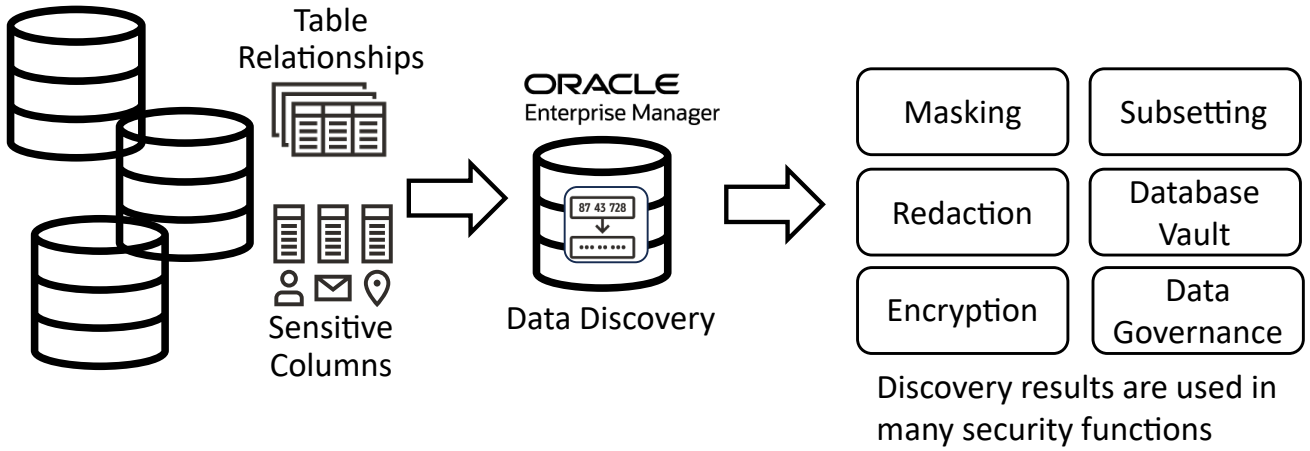


Figure 4-10: Results of data discovery are useful in configuring other security controls

Use cases and deployment scenarios

Enterprise Manager's discovery capabilities are particularly valuable in the following scenarios:

- **On-premises environments:** Organizations with data residency requirements or security policies that prohibit using cloud-based management tools can deploy Enterprise Manager entirely within their own data centers.
- **Integrated lifecycle management:** When organizations use Enterprise Manager for provisioning, patching, backup, and performance management, adding discovery and masking capabilities provides a unified operational platform.
- **Complex application environments:** Applications with intricate data relationships benefit from Enterprise Manager's sophisticated referential relationship discovery, reducing the risk of masking-induced application failures.
- **Test data management programs:** Organizations with formal processes for provisioning test and development environments can integrate discovery and masking into automated workflows that create compliant, functional nonproduction databases.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Database Security Assessment Tool
- Get Started with Oracle Data Safe Fundamentals
- Oracle Database Security Central
- Data Masking and Subsetting (including sensitive column discovery)

These workshops provide pre-configured environments with sample databases containing realistic sensitive data, allowing practitioners to execute discovery jobs, analyze results, and experiment with different configurations without impacting production systems.

Summary

Sensitive data discovery should be considered an essential foundational step, not an optional audit activity. Security controls are more effective when teams know where regulated, sensitive, and high-impact data resides. Many organizations invest in strong defenses but still lack a reliable way to locate the data that requires the greatest protection. By starting with discovery, teams can apply safeguards with purpose and precision.

The need is rooted in everyday business reality. Databases carry the information that keeps enterprises moving, and that same information often spans personal and sensitive categories like payment details, healthcare information, employee compensation, and proprietary metrics. Sensitivity also shapes adversary motivation: high-value data attracts high-intent threat actors, and exposure can cascade quickly into reputational, financial, and legal consequences. Practical direction boils discovery down to three questions every team should be able to answer: what types of sensitive data exist, how much exists, and where it resides. That clarity helps avoid a common failure mode, overprotecting low-risk areas while critical information remains exposed.

AI is positioned as a major amplifier of the exposure surface, especially when RAG systems and model integrations connect to production databases. Concrete scenarios make the risk tangible, such as an internal query that unintentionally surfaces executive compensation details. Regulated data used in AI training is also flagged as a long-tail risk, because it may later become recoverable through prompt injection, model inversion, or inference. MCP is described as a standard connector that speeds AI-to-database integration, which increases the urgency of putting guardrails in place early. In that world, discovery becomes the first control: define exclusions, row-level policies, masking or tokenization choices, and role-based access boundaries before AI-powered querying begins.

For execution, the recommended playbook is a blended approach: metadata-based scanning plus content-oriented pattern detection to balance speed, coverage, and confidence. Metadata discovery is fast because it relies on column names and comments, but it becomes brittle when naming practices are weak. Pattern-based discovery adds confidence by sampling values to detect formats and validate identifiers, though it comes at higher cost and demands careful handling of sampling results. These approaches map naturally to Oracle tools and their best-fit use cases: DBSAT for rapid assessments and reporting, Oracle Data Safe for broad discovery and evolving sensitive data models, Oracle Database Security Central for strengthening monitoring and firewall policies without sampling values, and Enterprise Manager for deep referential relationship discovery that supports masking and other downstream controls.

The next chapter covers Database Authentication, where discovery shifts from a map on the wall to a living checkpoint at the door. If this chapter clarifies what must be protected and where it hides, the next chapter lays out the mechanisms that decide who gets close to it, under what conditions, and with what proof, turning visibility into a disciplined, persuasive security posture readers can put into practice.

Chapter 5

Authenticating Database Users

The importance of strong authentication

An essential but often neglected principle in designing and implementing security solutions is that comprehensive identity-based security cannot exist in an anonymous environment. While this may seem self-evident, it is surprising how frequently this vital aspect is overlooked in database application design. Furthermore, applications sometimes obscure the user's identity from the database, which can undermine many of the built-in security features databases offer.

In virtually all robust database security frameworks, the ability to consistently identify users and authenticate them with strong, dependable methods forms the bedrock of the overall security model. Without this foundation, other security measures cannot be effectively enforced. This chapter helps you choose authentication methods, map identities to schemas, and operationalize secrets and accountability.

Unraveling identification, authentication and authorization

The database security process can be viewed as three key steps:

1. **Identification**: The user begins by presenting their identity to the database, such as by entering a username.
2. **Authentication**: The user then demonstrates that the presented identity is valid, such as by providing a password. The database verifies whether the entered password matches the one associated with the username.
3. **Authorization**: If the credentials are correct, the database accepts the user's identity as trustworthy. The database then assesses what privileges and authorizations are assigned to that user. Data security measures are then enforced based on these specific privileges and authorizations.

Most security efforts and resources are concentrated on processes within the third step, where fine-grained access and authorization controls are managed. Nevertheless, the importance of the first two steps cannot be overstated. Identification and authentication form the critical foundation of the security model.

This chapter explains how an authentication strategy helps protect database users and the data they access from attackers. It also provides guidance on managing user accounts, whether those accounts are maintained locally within the database or managed through centralized external services such as a directory service or a cloud identity provider.

The next chapter turns to user and application authorization, focusing on how to control what a user or application can do within the database.

Attacker tactics

One of the simplest and, regrettably, most frequent methods of breaching a database involves impersonating a legitimate user with authorized access. Attackers typically target accounts with elevated privileges, as these provide broad latitude for carrying out malicious activities. Robust authentication serves as a key defense against this form of attack, greatly lowering the chances of unauthorized access.

It is important to examine how attackers employ different tactics to obtain credentials in their efforts to access your data. Consider the various ways attackers attempt to gain access by defeating your authentication scheme:

Social engineering and phishing: Hackers often employ social engineering tactics to obtain account credentials, using spear phishing attacks that target end users or database administrators within an organization. These individuals are easily identified on platforms such as LinkedIn.

GenAI as a social engineering weapon: GenAI increases its success by making fraudulent communications more persuasive, more personalized, and easier to iterate at scale. The FBI has specifically warned about campaigns using AI-generated voice messages (vishing) and malicious text messages (smishing) to impersonate trusted individuals and trick targets into transitioning to attacker-controlled platforms or disclosing login information. As a result,

database authentication design should assume that credential theft and impersonation attempts will be more frequent and more convincing, reinforcing the need for strong authentication.

Hardcoded or exposed credentials in applications: Another risk area involves the discovery of hardcoded database connection details. Applications frequently store database usernames and passwords directly in program code or in plaintext configuration files. Since application servers are typically located closer to the network edge than database servers, they present an easier target for attackers. Once an application's service account is compromised, malicious actors can exfiltrate, alter, or erase any data that account is authorized to access. This applies equally to microservices and containerized environments where environment variables might be exposed.

Default or published passwords: Attackers may also attempt to use common default or well-known passwords to gain unauthorized access. Successful entry allows them to exploit the privileges associated with those user accounts and access sensitive information.

Brute force password attacks: When passwords aren't complex, and there's no retry limit, attackers can use automation to try millions of combinations until one works. Oracle AI Database makes this much harder by using exponential backoff. Each failed attempt increases the waiting time before another login try. This slows attackers down, but if simple passwords are allowed, brute force attacks might still succeed.

Credential stuffing: Credential stuffing is a variation of brute force attacks, where account information from previously breached sites is reused in attempts to compromise additional systems. The prevalence of password reuse increases the effectiveness of this strategy, with vast collections of stolen credentials available on the dark web. Automated tools can rapidly test millions of these credential pairs against your organization's systems. If attackers have already identified valid usernames, searching stolen databases for associated passwords allows them to target your infrastructure more effectively.

These techniques aren't especially sophisticated, and AI-automated tools make them easy to run. It's important to emphasize, if attackers sign in to a compromised account, they gain access to everything that account can access. Strong authentication reduces that risk, and account hygiene ensures you keep exposure low.

Database authentication methods

The attacks listed above share a common trait: they all focus on exploiting password-based authentication. Compromised credentials are at the heart of most successful security breaches, prompting many organizations, including the US federal government and Oracle, to adopt stronger authentication methods such as tokens, passkeys, and biometrics. Implementing multi-factor authentication (MFA), which relies on two or more independent factors, enhances security by ensuring a single compromised factor does not jeopardize the entire account.

Instead of checking passwords against the database's internal hashed credential store or a centralized directory, Oracle AI Database can authenticate through external services. You can use OCI Identity and Access Management (IAM) and Microsoft Entra ID, Kerberos, public key certificates, and RADIUS. These options help you adopt phishing-resistant, standards-based authentication while keeping administration centralized. A new feature introduced with the July 2025 release update lets you require approval via a mobile authenticator, such as Oracle Mobile Authenticator or Cisco Duo, for password-authenticated local database accounts – extending the benefits of MFA even to those local accounts.

After authentication, the user is mapped to a database schema consisting of tables, views, indexes, and procedures and then granted appropriate authorization through roles and privileges. In early Oracle Database releases, the user account and schema were the same, and each user had a dedicated schema. That heritage remains in the command syntax: you create a schema with CREATE USER. There is no CREATE SCHEMA command. A specialized form of CREATE USER creates a schema-only account - a user with no authentication credentials. When you use local database password authentication, the schema and user remain the same entity. When you authenticate with a

directory or identity service, we map identities to schemas. You can assign a private schema to each person or map multiple people to a shared schema.

While the topic of authorizations is covered elsewhere in the book, it's useful to note that roles - collections of privileges - can be managed in the database or, in some cases, managed externally through a directory service, OS groups, or a RADIUS server.

Table 5-1 lists common database authentication methods and how they map to schemas and roles.

Authentication Method	Schema Mapping	Role Mapping
Database Password	Schema and user are the same entity	Managed in the database
Operating System	Database maps the user to a schema	Managed in database or through membership in OS groups
Kerberos	Database maps the user to a schema	Managed in database
Public Key (PKI)	Database maps the user to a schema	Managed in database
RADIUS	Database maps the user to a schema	Managed in database or through RADIUS server
Oracle Directory Services	Managed in database and directory service	Managed in database or through membership in directory service groups
Microsoft Active Directory	Managed in database and directory service	Managed in database or through membership in Active Directory groups
Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM)	Managed in database and identity service	Managed in database or through membership in OCI IAM groups
Microsoft Entra ID	Managed in database and identity service	Managed in database or through membership in Entra ID groups

Table 5-1: Database user authentication and authorization methods

As noted above, despite the inherent challenges of passwords, they remain the most widely used authentication token. When database users authenticate with a password, it is important to ensure passwords are tightly managed and made as secure as possible. Below are a few areas to review when passwords are used for authentication.

Managing passwords & secrets

With password-based authentication, people can type a password to connect to the database, but applications, middle-tier systems, and batch jobs cannot rely on a human at the keyboard. In the past, many teams embedded usernames and passwords in code or stored them in clear-text configuration files. That approach expanded your attack surface and forced you to change scripts whenever a password changed.

Secure External Password Store (SEPS) improved on that model by storing credentials, including passwords, private keys, and certificates, in an encrypted Oracle Wallet. With SEPS, you only needed to remember the wallet password. Once unlocked, the wallet could be used by applications or servers to access multiple databases securely, including with auto-login wallets for 24/7 operations. Wallets remove clear-text passwords from command-line history and configuration files, making casual discovery harder for anyone with access to the operating system.

However, SEPS wallets still lived on the application server or client workstation and were protected only by the wallet passphrase. If an attacker stole the wallet files, they could try to brute-force the passphrase or simply wait for a compromised host to capture the wallet password when someone used it.

Secrets management offers a more secure, modern way to store and distribute passwords. With a secrets manager such as Oracle Key Vault, users and applications retrieve credentials through an API call. The password is centrally managed and is not stored on the application server or user workstation.

A secrets manager, like Oracle Key Vault, has the following benefits:

- Centralize credentials instead of maintaining SEPS wallets on many servers.
- Simplify password rotation and remove persistent password footprints from application hosts.
- Reduce operational overhead compared to provisioning and protecting local wallets at scale.

For applications that still use password-based authentication, a secrets manager is the recommended approach for storing and distributing credentials. We'll dive deeper into Oracle Key Vault's role in encryption and key management in Chapter Eleven.

Accounts and passwords: to share or not to share

When it comes to accounts and passwords, the best practice is obvious: do not share them among users and administrators. When multiple people share a single account, accountability quickly breaks down, and the result is a security nightmare. If errors occur, sabotage is suspected, or data is stolen, it is nearly impossible to pinpoint which individual behind the shared account is responsible. Most organizations even have policies that prohibit shared accounts, yet those rules are still routinely ignored, sometimes by the most highly privileged database users, notably developers and database administrators.

In practice, database administrators often connect through shared accounts, most commonly the database super user (SYS) or the default SYSTEM database administrator account. These accounts are powerful and should not be used for day-to-day DBA activities. A more defensible approach is for each DBA to have a personal account granted only the roles or privileges required for the role, whether that is the out of box DBA role or a tighter role tailored to the tasks the DBA performs.

The same guidance applies to application administrators and developers. Maintenance work may require access to an application schema, but that does not mean the application service account credentials should be used. Later in this book, methods for blocking misuse of service accounts using Database Vault are discussed. For now, the priority is to enable administrators and developers to work through their own private accounts in a way that remains practical and low friction.

Proxy authentication solves this. It lets an administrator act as the application account without knowing the application account's password. With proxy authentication, a person signs in with their own account and then acts as another user. Audit records still show the real end user, and access control policies can consider both the true identity and the proxied account. That accountability is the key benefit for operations and compliance.

When you authorize proxy access to an application account, administrators authenticate to the database with their own credentials and then proxy to the application schema without needing the schema's password. For example, in Figure 5-1 we grant alice_appdba the CONNECT THROUGH privilege to connect as hrapp. This approach keeps credentials secure, preserves least privilege, and gives you clear audit trails for every action.

```
SQL> ALTER USER alice_appdba
GRANT CONNECT THROUGH hrapp;
```

Figure 5-1: Example showing how to grant proxy privileges

`alice_appdba` then connects using their password and assumes the identity and privileges of the `hrapp` schema by proxy as shown in Figure 5-2:

```
SQL> CONNECT alice_appdba[hrapp]
Enter password: <alice_appdba_password>
```

Figure 5-2: Example showing how to connect using proxy privileges

With this setup, audit records show `hrapp` as the `DBUSERNAME` and `alice_appdba` user as `DBPROXY_USERNAME`. The proxy username is also available for policy-driven access control mechanisms like Database Vault, Label Security, Data Redaction, and Real Application Security (discussed later in this book).

Increase password strength

There are many situations where passwords remain the only practical option for authentication. Older applications and clients often do not support other authentication methods. In those cases, the goal is to make passwords as strong as users and applications can reasonably support and to apply solid password discipline, such as avoiding password reuse, avoiding shared accounts, and managing the password lifecycle.

Like most systems that support password authentication, Oracle AI Database provides controls for minimum password length, complexity, and lifecycle. These controls are managed through user profiles. Every database account is assigned to a user profile. In theory, each database user could have a unique profile, but that is uncommon. More often, a small set of profiles is used, with users grouped under different profiles based on their needs. User profiles define password and resource authorization parameters for all accounts assigned to them. If no profile is specified when a user is created, the default profile is automatically assigned. Custom profiles can be created as needed

Note: Oracle AI Database 21c and earlier support passwords up to 30 bytes. Beginning with Oracle AI Database 26ai, the maximum password length is 1024 bytes.

Typically, the default profile is tailored to meet organizational standards, and other profiles are reserved for situations that require an approved exception. A profile can contain up to 18 parameters, with nine parameters dedicated to resource limits and nine dedicated to authentication. When a new profile omits any of the 18 parameters, the unspecified settings automatically inherit the values defined in the default profile.

Best practice for interactive accounts: For accounts used by people rather than applications or batch processes, passwords should be supplemented with the local user MFA feature. A second factor, such as a mobile authenticator approval, should be required for local database logins to reduce the risk of credential stuffing and brute force attacks. This approach strengthens human sign-ins without changing how service accounts authenticate.

Figure 5-3 shows an example custom profile, app_profile, that includes both authentication and resource parameters. This custom profile specifies six of the 18 possible parameters - the remaining 12 inherit values from the default profile.

```
SQL> CREATE PROFILE app_profile LIMIT
PASSWORD_VERIFY_FUNCTION ora12c_stig_verify_function   -- STIG password complexity rules
PASSWORD_ROLLOVER_TIME 30               -- Allows both old and new passwords for 30 days
FAILED_LOGIN_ATTEMPTS 6                 -- Lock the account after 6 attempts
SESSIONS_PER_USER 2                     -- Allow two sessions for each user
IDLE_TIME 30                            -- Automatic logout after 30 min idle
PASSWORD_LIFE_TIME 180;                 -- Force password change after 180 days
```

Figure 5-3: Sample password profile

All eighteen parameters can be important, but let's call out three that can make the biggest impact on your security:

`PASSWORD_VERIFY_FUNCTION:` This enforces strong password complexity through built-in functions like ora12c_stig_verify_function, which aligns with U.S. Department of Defense Security Technical Implementation Guide

(STIG) recommendations. These rules require a minimum length, a mix of alphanumeric and special characters, and more. If Oracle's default functions aren't enough, you can write your own; just start with the sample code in `$ORACLE_HOME/rdbms/admin/catpvf.sql`.

`PASSWORD_ROLLOVER_TIME:` Application accounts often require periodic password changes to stay compliant with security policies. In the past, updating these passwords meant juggling downtime and reconfigurations to avoid failed logins. With Oracle Database 19c and above, you can update a password for an account but keep the old one working for a set period (the rollover time). This lets you update all application clients smoothly. No downtime or risk of service disruption. Set this parameter for application service account profiles but leave it off for human users. If you use zero, password rollover is disabled.

`FAILED_LOGIN_ATTEMPTS:` This parameter sets how many consecutive failed login attempts are allowed before an account is locked. It's a simple, effective way to block brute-force attacks.

Taking these steps helps you enforce your organization's standards, make password management easier, and keep your users and data safer, no matter which type of authentication you use.

Simplify account administration

Most databases have only a small number of human actor accounts, typically database administrators, perhaps a developer or two, and a few power users who run reports. Many real-world environments, however, break that pattern, including databases with tens of thousands of valid human actor accounts. Even when there are only a few users per database, scale becomes a factor when those same accounts exist across many databases, and customers with thousands of Oracle AI Databases are not unusual.

In either case, managing identities inside the database can become an operational burden. People join and leave, and roles change. Some users no longer need database access, while others should have different or fewer privileges. At scale, accounts that are no longer needed often remain active long after a user has left or changed roles, creating attractive targets for hackers seeking unauthorized access to data.

Oracle AI Database supports centralized user management by allowing users and credentials to be managed outside the database, most commonly in a directory such as Microsoft Active Directory or a cloud identity service such as Oracle Cloud Infrastructure's Identity and Access Management (IAM), Okta, or Microsoft Entra ID.

Centralized user management ensures consistent policy enforcement across both users and databases. It also streamlines the process of adapting to organizational changes as individuals join, depart, or take on new roles. Many centralized services, especially cloud identity providers, can also require multi-factor authentication. Centralization remains valuable even beyond passwords. For example, Microsoft Active Directory integrations more commonly use Kerberos for authentication. PKI certificates, often delivered through smart cards like the US Department of Defense Common Access Card (CAC), are also widely used.

We'll cover centralized user management in more depth in Chapter Six.

As applications, tools, and databases continue shifting to the cloud, multicloud identity integration becomes essential. It allows people to be managed in one identity service while their applications, databases, and tools operate across multiple clouds or on premises. Chapter Fourteen explores multicloud identity in more detail.

Protect user accounts

It is not realistic to rely on (business) users to remember, interpret, and voluntarily follow security guidance on password strength, password management hygiene, and appropriate roles and privileges. The database should enforce these rules, so they are applied consistently and predictably. Policies such as the following help protect user accounts from compromise:

Avoid password authentication. Use strong authentication mechanisms such as cloud identity service-provided tokens, Kerberos, RADIUS, or PKI certificates.

When passwords are unavoidable, apply a strong password profile to govern password strength, inactivity period, and the number of password retries (failed logins).

Centralize user management with an identity service like Active Directory to reduce the likelihood of orphan accounts when someone leaves the organization or changes roles.

This best practice shifts the burden from people to the database and your identity platform, improving security without adding unnecessary friction.

Protect DBA accounts

Because database administrator accounts typically carry broad access, they are prime targets for attackers seeking a high impact compromise. DBA authentication should follow these best practices:

- Require named DBA accounts with strong authentication to preserve accountability. Prohibit shared accounts and default accounts such as SYSTEM. Where practical, use strong authentication such as cloud identity service tokens, PKI certificates, or Kerberos.
- Use a Privileged Account Management (PAM) system for root and database owner operating system access needed for SYSDBA, typically limited to upgrades, patching, and database restarts.
- Use sudo when DBAs must access the database operating system account, so activity is recorded and attributable to an individual user.

If DBAs log into the database server operating system before connecting to the database:

- Require access through a personal named account, not the account that owns the database binaries.
- Provide a separate client such as Oracle Instant Client or SQLcl, rather than granting access to the database owner account or binaries.

When DBA responsibilities are split into specialized teams, such as backup and recovery DBAs, performance tuning DBAs, and security managers, avoid granting the default DBA role. Create specialized administrator roles for each function and grant only the privileges required for that team (more about this in Chapter Six and Chapter Seven).

Protect application accounts

Application service accounts in the database not only hold the roles and privileges required to run the application on behalf of every application user, but for convenience they may also be granted additional roles and privileges for installation, upgrades, patching, and other maintenance. Because these accounts can become powerful keys to the environment, they should be protected using the following best practices:

- Use strong authentication mechanisms (cloud identity service tokens or PKI certificates) for database access.
- Use controls such as SQL Firewall, Database Vault connect rules, or logon triggers to limit where service account connections can originate. Additional discussion of Database Vault and SQL Firewall appears in Chapter Seven (for Database Vault) and Chapter Eight (for SQL Firewall).
- Use a secrets manager such as Oracle Key Vault for credential storage, and have applications retrieve credentials via API calls. Avoid embedding credentials in code or storing them in clear text properties files on the application server. Application servers are often exposed to a higher risk of compromise than database servers because they are typically deployed in network zones with a broader threat surface.
- Use gradual password rollover to reduce the risk of outages during password rotation. This allows both the old and new passwords to remain valid until updates are complete across all application servers.

- Require application administrators to use proxy authentication instead of logging in with the application service account.

Protect security administrator accounts

Security administrators manage security controls, encryption keys, database users, and audit records. They protect the database and set user constraints on tablespaces, idle time, and concurrent sessions so one user doesn't impact others. Use the following practices to strengthen accountability and reduce risk.

Use strong authentication. Authenticate with PKI certificates, Kerberos, or tokens. For interactive logins, add a second factor with local user MFA to reduce credential stuffing and brute force risk.

Enforce separation of duties. Keep security administrator responsibilities separate from DBA and general user duties. Do not allow DBA or user accounts to alter security records, create fake users, or change security controls. Use roles and policies that restrict who can manage users, keys, and audit configurations.

If you must use passwords, enforce strong profiles. Assign security administrators to profiles that require strong passwords, set inactivity and expiration rules, and automatically lock accounts after too many failed login attempts. Use profile settings to enforce idle time limits, concurrent session caps, and tablespace quotas.

These controls create clear accountability, reduce the blast radius of compromise, and help you maintain consistent security across your databases.

Simple designs yield strong security

One best practice is as much about architecture as it is about requirements. The simple pattern is to create a schema only account for the application objects and procedures so the application schema cannot be used for direct logins. Then use a separate run time application service account to access those objects through controlled procedures and views.

Schema only accounts were introduced in Oracle AI Database 18c, providing a named schema with no password or other ability to log in. Starting with Oracle AI Database 19c, almost all Oracle accounts installed with the database are schema only accounts to prevent login to these privileged accounts. In earlier versions, these accounts had passwords that required periodic rotation. Other users can still be granted read and write access to these schemas**.**

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- DB Security – Privilege Analysis
- Proxy authentication
- Centralize database user management with OCI IAM

Summary

Authentication often sits quietly at the critical center of database security, and it's easy to underestimate. Security only becomes real when it's clear who is present. When end-user identity is blurred or hidden, built-in database safeguards can lose much of their power. A practical way to frame the journey is in three steps: identification, authentication, and authorization. The first two steps do the foundational work that makes the third enforceable. Solid programs account for both locally managed database accounts and externally managed identities delivered through directory services and cloud identity providers.

Attackers fixate on authentication for a simple reason: impersonation remains one of the easiest, most reliable ways into an environment, and privileged accounts can turn a foothold into a wide-open corridor. Credential capture and replay show up in many forms, from spear phishing and social engineering to GenAI-fueled vishing and smishing. The risk also includes exposed application secrets, default passwords, brute force attempts, and credential stuffing at industrial scale. The sobering reality is that once an attacker signs in as a legitimate account, they inherit everything that account can reach. The most effective response is to reduce the odds of a successful sign-in and keep exposure low through disciplined account hygiene.

That's why modern guidance pushes teams to move beyond password-centric design toward stronger authentication methods and multi-factor authentication. Oracle AI Database support for external authentication is positioned as a key enabler here, with options including OCI IAM, Microsoft Entra ID, Kerberos, public key certificates, and RADIUS. A July 2025 update is also highlighted for extending mobile authenticator approval to local password-authenticated accounts, bringing MFA benefits into scenarios that traditionally lag. From there, the model stays grounded: authenticated identities are mapped to schemas, and roles and privileges can be managed inside the database or through external systems, depending on the chosen authentication method.

Operationally, secrets storage and account management are treated as first-class controls, not cleanup work. Credentials embedded in code or stored in plaintext files expand the attack surface. Wallets like SEPS can reduce casual discovery, but they're still vulnerable if stolen. A more modern model is centralized secrets management, with tools such as Oracle Key Vault providing API-based distribution and control. Shared accounts, especially for DBAs and other powerful users, are called out as a risk to accountability; the recommendation is personal named accounts with least privilege. Proxy authentication is presented as a practical way to preserve accountability and audit clarity while avoiding direct use of service account credentials. Password profiles and key parameters round out the operating playbook, including complexity verification functions aligned with STIG guidance, password rollover time to smooth application transitions, and failed-login lockouts, along with an implementation note that Oracle AI Database 26ai supports passwords up to 1024 bytes.

After tightening authentication, a key truth remains: many database compromises look like a normal login that never should have worked, followed by valid commands that were never meant to be available. The next chapter, Database Access Controls, therefore, emphasizes precision over perimeter thinking by defining who can act on which data, with what scope, and why, even for a DBA, an application service, a developer, or an analyst. Through privileges (object, schema, system, and administrative) and role design, authorization becomes a deliberate discipline. Unnecessary grants expand the blast radius, while well-chosen constraints provide guardrails under pressure. For resilience against mistakes, misuse, and attackers using legitimate credentials, this is where the book becomes a practical playbook. Database Access Controls is the next destination, and it is where thoughtful authentication choices start to pay off by letting policy guide action with precision.

Chapter 6

Controlling Database Access

Access control: A foundation for data security

User impersonation is one of the most common and effective methods attackers use to steal, alter, or destroy data. This is precisely why controlling what users can do is essential to protecting data. Every unnecessary privilege granted to a user increases risk to the data and the database, whether the account is compromised or the user acts in ways that violate organizational policy.

Keep access control central

Access controls define who can act on which data and what actions they can perform within the database. Limiting each account to only the capabilities required for its intended role reduces risk. Overprivileged accounts are a serious and persistent threat to data security. Strong access controls also support regulatory compliance and help reduce the risk of data theft, destruction, and misuse.

Because access control is so critical, Oracle AI Database provides many ways to manage and restrict access. In some cases, the differences between access control features are obvious. In others, the distinctions are subtle, and it is helpful to use this chapter and the next few chapters to identify the right feature or tool to meet a specific objective.

The discussion starts with the basics of privileges and roles, and then moves into more advanced topics in later chapters. It also covers managing user account authorizations locally and through directory or identity services.

Database privileges

A database privilege grants the ability to perform specific operations on data objects or to execute certain statements. Oracle AI Database has four privilege types: object, schema, system, and administrative. The first three types provide progressively broader scopes of authority.

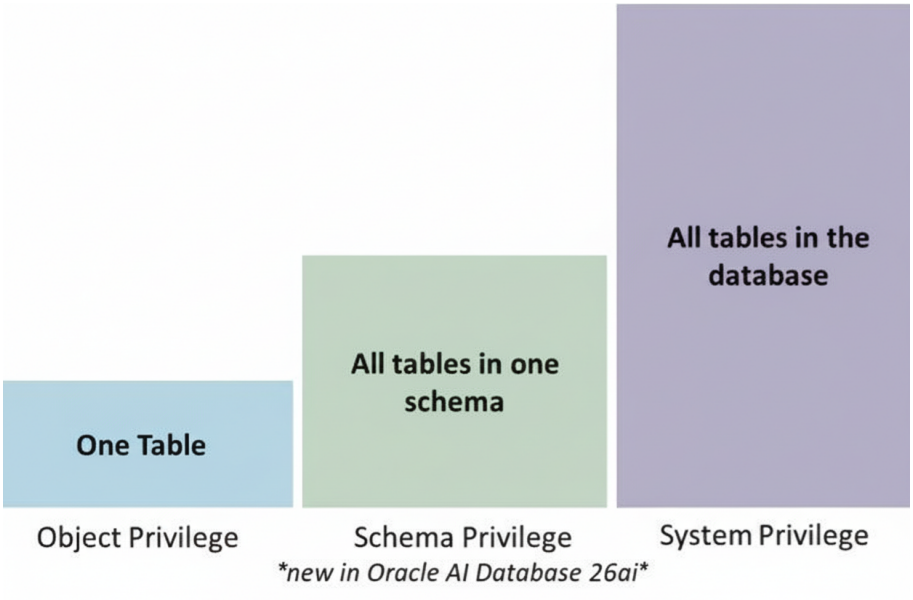


Figure 6-1: Summary illustrating the scope of object, schema, and system privileges

Object privileges are fine-grained permissions that allow a user to perform specific actions on a particular database object. Examples include querying data from a table (SELECT or READ), modifying data (INSERT, UPDATE, DELETE), changing the structure of a table (ALTER), or running stored program units such as procedures or packages (EXECUTE).

To grant user Scott the object privilege to select data from the customers table in the app schema, use the following command:

```
GRANT SELECT ON app.customers TO scott;
```

Figure 6-2: Example syntax for granting an object privilege

Schema privileges, introduced in Oracle AI Database 26ai, are designed for applications and users that need to manage objects in another schema.

As a general rule, schema privileges are preferred over system privileges because they are limited to a specific schema. That tighter boundary makes them more secure than system privileges. They are also easier to maintain than granting individual object privileges, since there is no need to rework grants every time a new object is added to a schema.

The following command grants user Scott the ability to select data from all tables or views in the schema owned by app.

```
GRANT SELECT ANY TABLE ON SCHEMA app TO scott;
```

Figure 6-3: Example syntax for granting a schema privilege

System privileges provide broader access to database objects and typically to all objects relevant to the privilege.

For example, `SELECT ANY TABLE` allows a user to read data from almost any table in the database, including sensitive data stored in application schemas. `CREATE USER` is another system privilege, enabling the creation of new users in the database.

In most situations, system privileges should not be granted to non-administrators due to their expansive scope.

To grant the powerful `SELECT ANY TABLE` privilege to user Scott, use the following command:

```
GRANT SELECT ANY TABLE TO scott;
```

Figure 6-4: Example syntax for granting a system privilege

Administrative privileges differ from object, schema, and system privileges. They are centered on database administration tasks rather than access to database objects. Administrative privileges are commonly used for activities such as database backup, encryption key management, and core database operations such as startup and shutdown.

Examples of administrative privileges include `SYSDBA`, `SYSOPER`, `SYSKM`, and `SYSDG`. The syntax for granting an administrative privilege closely resembles an object privilege grant. For example, the following statement grants Scott the `SYSKM` privilege, which provides the ability to manage encryption keys for the entire database:

```
GRANT SYSKM TO scott;
```

Figure 6-5: Example syntax for granting an administrative privilege

Certain maintenance activities, such as database upgrades and patching, can only be performed by the SYS account, the database owner. The SYSDBA administrative privilege allows a user to become SYS for these tasks. The SYS account and the related SYSDBA privilege should be used only when necessary and carefully safeguarded. As a best practice, fully audit all use of administrative privileges.

Each administrative privilege is associated with an operating system group, such as OSBACKUP or OSKM. Depending on organizational structure, these privileges may be assigned to a single shared group, such as DBA, or each privilege may be mapped to its own dedicated operating system group.

Administrative privileges are extremely powerful. Even when granted to a user, they are not active within a session by default. The user must explicitly activate the granted privileges when starting a database session.

```
CONNECT scott@pdb1
-- Scott's SYSKM privileges are not activated and cannot be used during this session
CONNECT scott@pdb1 AS SYSKM
-- Scott's SYSKM privilege is active in this session and Scott can perform encryption key management operations
```

Figure 6-6: Activating an administrative privilege within a database session.

Database roles

Roles are named groups of privileges. They are used to streamline privilege management by making it easier to grant or revoke the same set of privileges for multiple users. Object, schema, and system privileges can be bundled into task-based roles. Roles can also be organized hierarchically under other roles. Granting roles to other roles lets you group privileges into a task role, and multiple task roles can then be combined into a broader organizational role.

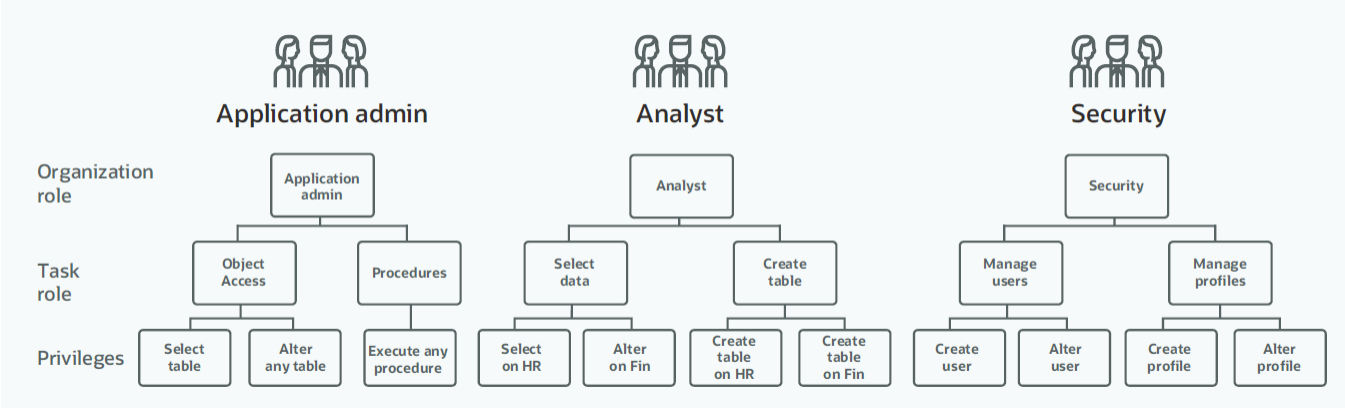


Figure 6-7: Example three-tier role and privilege hierarchy

A single user can be granted multiple database roles. This model simplifies onboarding by granting the role for a position instead of identifying and assigning each required privilege individually. As responsibilities shift between groups, task roles can be moved to match the new structure rather than managing individual privileges or redefining organizational-level roles. Roles therefore simplify privilege management during organizational change.

Types of database users

Users receive privileges either by direct grants or through a granted role. In general, a user can connect to their schema and access, modify, or delete all objects within that schema. Through privileges and roles, users can also be granted permission to perform specific operations, such as updating an object in another schema, or database specific rights, such as the ability to export or import data. These users may be human actors (individual users), applications, AI agents, tools, automated processes, or other programs acting on behalf of users.

A core security principle is to grant only the privileges needed to do the job. The challenge is that needed privileges vary based on what the account is meant to do. For practical access control planning, it helps to group accounts into categories based on how they are used and what they will do in the database:

- **Database administrators (DBAs)**: Every database relies on one or more administrators to handle tasks such as performance management, diagnostics and tuning, upgrades and patching, startup and shutdown, and backup. Unless access is constrained, DBAs can typically view and modify all data in the database, including sensitive personal, health, and corporate finance records, even though that level of access is not required for many DBA tasks. That broad access also makes DBAs a prime target for hackers.

- **Security administrators**: Many organizations have specialized DBAs who concentrate on security work, including user account management, access control policy development and implementation, encryption key management, and audit management. Because their work is narrower than a general-purpose DBA, these accounts usually need fewer privileges. For example, a security administrator generally does not need to create schema objects or access application data. This kind of specialization is common in large organizations with many people and many databases. One team may cover backup and recovery, another performance tuning, and another patching and upgrades. Each team needs only a subset of DBA privileges rather than the full DBA role.
- **Application service accounts**: These are called service accounts (sometimes described as "robotic", "non-interactive", or "machine" accounts) because they connect to the database without a human directly initiating the connection. Most databases have applications, business intelligence tools, or reporting systems that connect using one or more service accounts. If one of these accounts is compromised, the impact can be data loss across the entire application. Since applications are often built for high availability, service account passwords may be stored on multiple middle-tier servers, which increases compromise risk. And because no person needs to remember or type the password, it should be as long and complex as the database can support.
- **Application administrators**: These accounts manage, patch, and upgrade the application itself and usually have full access to the application's data and stored procedures. They tend to have DBA-like access to their application's data, but they do not need the broad database management privileges required by a DBA.
- **Developers and testers**: These accounts create and modify the database objects an application needs. Traditionally, developers did not have routine production database access. Changes were built in lower environments and promoted through defined change management processes. In DevOps or SecDevOps environments, those lines are often blurred, and developers with production access are now common. Even without production access, developer systems are frequently cloned from production and may contain the same data. Developer accounts typically have privileges similar to the application service account and may use proxy authentication to access the application schema. Oracle AI Database 26ai introduces a new DB_DEVELOPER_ROLE that includes the privileges a developer needs for their job function. Like DBAs, developers are often easy to identify through social media and can be targets for social engineering attacks.
- **Data analysts or business intelligence users**: Many databases support data analyst access, either directly or through a business intelligence tool. These accounts typically need read-only access to the application schema. They can pose slightly higher risk than users who connect through the application because they bypass application-level access controls.
- **Other database users**: This includes everyone else with a database account, such as batch processes, integration services like Oracle GoldenGate, or tools like Data Safe, Enterprise Manager, or Oracle Database Security Central. Their access requirements can vary widely, and many tools include scripts that grant what the tool considers appropriate access. It is worth treating appropriate with caution. A vendor may assume their tool needs the DBA role, but *Privilege Analysis* often shows that far less access is actually used. These accounts may be limited to their own schema or may need broader access to multiple schemas.

Auditing user privileges

One of the most critical components of an Oracle AI Database is the data dictionary, a set of tables and views that describe the database, including metadata for all objects and users. The dictionary views shown in Table 6-1 help identify the roles and privileges granted to users or to roles.

Dictionary views	Contents
DBA_TAB_PRIVS	Object privilege grants to roles or users
DBA_SCHEMA_PRIVS	Schema privilege grants to roles or users
DBA_SYS_PRIVS	System privilege grants to roles or users
DBA_ROLE_PRIVS	Roles granted to all users and roles
DBA_ROLES	All defined roles
ROLE_ROLE_PRIVS	Roles granted to other roles
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Object privileges granted to roles
V$PWFILE_USERS	Administrative privileges granted to users

Table 6-1: Data dictionary views for database roles and privileges

A close look at these dictionary views quickly highlights just how powerful the default DBA role is. It carries more than 200 system privileges, including `ALTER SESSION`, `CREATE`, `ALTER`, and `DROP USER`, `CREATE` and `ALTER ANY TABLE`, and `SELECT`, `INSERT`, `UPDATE`, and `DELETE ANY TABLE`. On top of that, it includes the EXPORT and IMPORT FULL DATABASE roles and more than a dozen additional roles, each with its own collection of privileges.

For organizations that use specialized DBAs, where each role only needs part of the full DBA capability, a better approach is to define one or more custom DBA roles, such as ops_dba or backup_dba, that match the actual job function. Privilege Analysis can help identify which privileges are truly required and shape these custom roles accordingly.

To determine a database user's complete set of privileges, it is necessary to review role grants, system privilege grants, and object privilege grants to build a full and accurate picture. The example below is taken from Oracle Data Safe's User Assessment module.

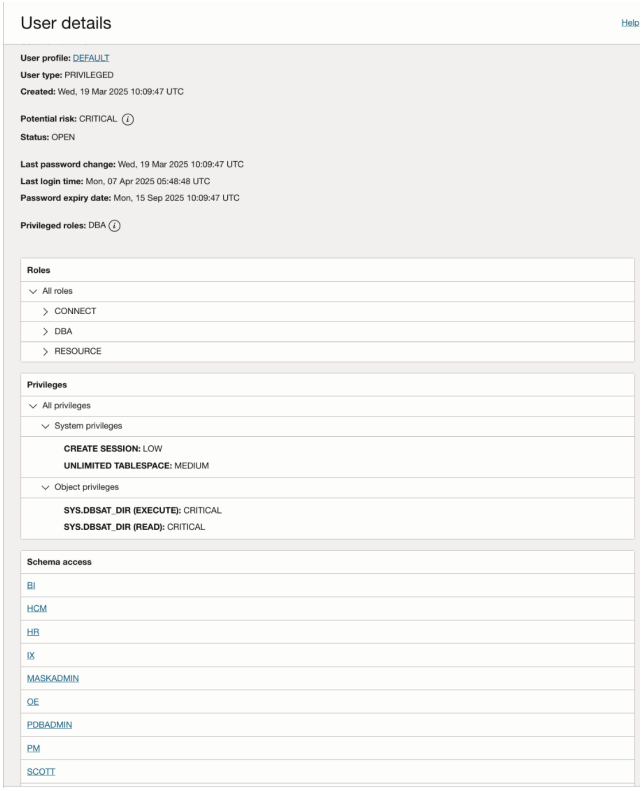


Figure 6-8: Example analyzing a user's privileges with Data Safe

Reviewing privileges in Oracle Database Security Central (Security Central) provides similar visibility into user and role access:

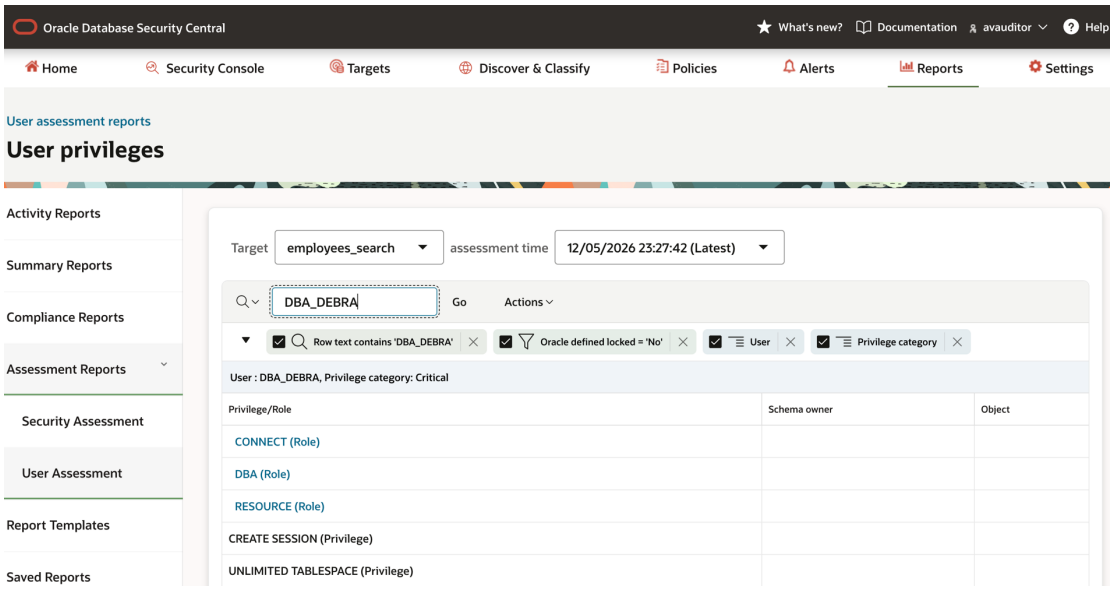


Figure 6-9: Example analyzing a user's privileges with Oracle Database Security Central

Centrally managed users

In enterprises where many users access multiple databases, it can be difficult for users to keep passwords compliant with each database's policy and to remember different passwords across accounts. Administrators also face significant overhead, as each user must be provisioned separately in every database along with a password. More importantly, each account must be deprovisioned when a user changes roles or leaves the organization. Accounts that are not deprovisioned after they are no longer needed are known as orphaned accounts. Attackers often exploit orphaned accounts to gain unauthorized access to data.

Enterprise User Security (EUS) centrally manages users and roles across multiple databases using an Oracle directory service such as Oracle Internet Directory or Oracle Unified Directory. The Oracle directory can run as a standalone LDAP server or integrate with other corporate directories. After a user authenticates successfully, the database consults the directory for authorization information, including roles. Database users managed through a directory service are known as enterprise users because they can span multiple databases across the enterprise. Enterprise users can be assigned enterprise roles or groups that determine access privileges across multiple databases. Enterprise roles in the directory map to one or more roles in the databases. Enterprise User Security is deprecated as of Oracle AI Database 26ai.

Centrally Managed Users (CMU) integrates the database directly with Microsoft Active Directory. Active Directory users can have their own schema or share a schema through membership in Active Directory groups. Active Directory groups can also map directly to database roles. One key distinction between EUS and CMU is where schema and role mappings can be defined. With EUS, mappings can exist in either the database or the directory. With CMU, schema and role mappings are always maintained in the database.

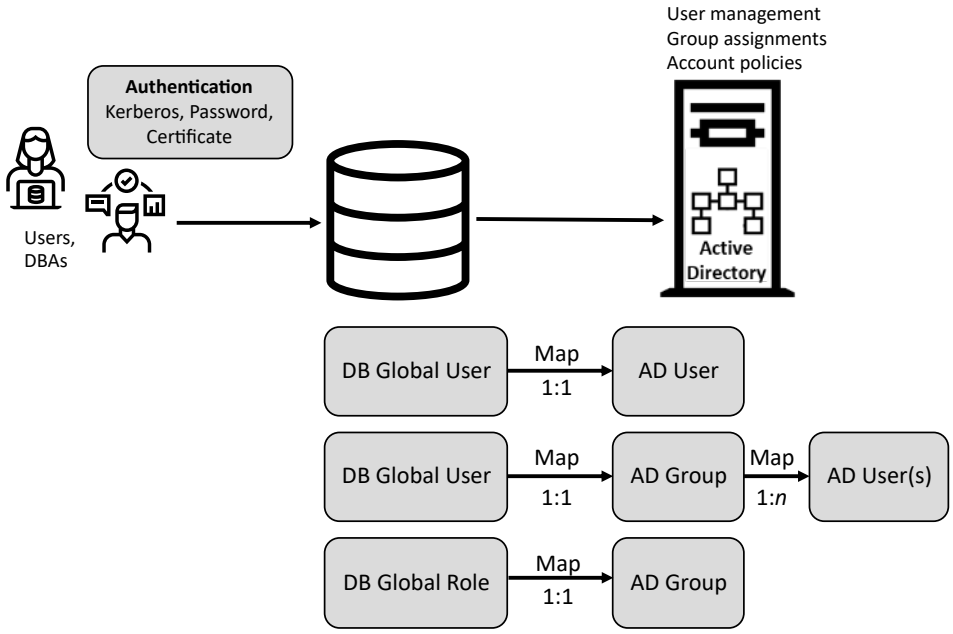


Figure 6-10: Illustrative example of Centrally Managed Users (CMU) mapping database global users/roles to AD users/groups

EUS and CMU allow users to authenticate using passwords, Kerberos, and Public Key Infrastructure (PKI) certificates. Kerberos is commonly used for human users, while passwords and certificates are typically used for application accounts.

Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users can access the database using IAM tokens. Similar to CMU, authorization is based on mappings to database schemas and roles. IAM users can map exclusively (1:1) to database schemas, or IAM users in an IAM group can be mapped to a shared database schema (many:1). With a shared schema, IAM security administrators can add an IAM user to that shared database schema by adding the user to the appropriate IAM group. Shared schemas remove the burden of creating, managing, and dropping database schemas for every IAM user. IAM groups can also be mapped to database roles, allowing IAM users to receive privileges and roles through membership in different IAM groups. IAM authentication and authorization is available for all OCI databases, including Autonomous Database, Oracle Base Database Service, and others.

Microsoft Entra ID users can access the database using Entra ID tokens and can use Entra ID app roles for database authorization. App roles are included in the Entra ID token and are used by the database to map Entra ID users to a database schema and, optionally, database roles. Entra ID authentication and authorization is available for Oracle AI Database 19c and higher, including databases deployed in OCI, in third-party clouds or on-premises environments.

Protecting database accounts

Protecting DBA accounts

Database Administrator (DBA) accounts typically have broad access within the database, often far beyond what is required to administer it. This access should be controlled using the following best practices:

- Grant only the privileges required for individual tasks rather than assigning the DBA role. If additional privileges and roles are needed for troubleshooting, revoke them after the task is complete.
- Use task-specific administrative privileges (for example, SYSKM or SYSBACKUP) instead of the SYSDBA privilege.
- Block access to user schemas or SQL commands using Database Vault.
- Fully audit all DBA activities for accountability and tracking.

Protecting security administrator accounts

Security administrators manage security controls, encryption keys, database users, and audit records. They manage the database's security posture and ensure there are appropriate constraints on authentication, failed logins, idle time, and the number of concurrent sessions. Security administrator accounts need to follow these best practices:

- Limit the privileges and roles granted to security administrators so that they don't have wide access to the sensitive data in the database. It would be unusual to grant a security administrator the DBA role.
- Fully audit all security administrator activity for accountability and tracking.

Protecting application service accounts

Application service accounts typically hold the roles and privileges required to run the application on behalf of every application user. For convenience, they may also include privileges for application installation, upgrades, patching, and other maintenance tasks. These accounts should be protected using the following best practices:

- Revoke the privileges for application upgrade, patching, and other maintenance activities from the application runtime account. Instead, create a separate administrator user that is managed and audited independently. Without this separation, a hacker who has compromised an application user account can use SQL injection attacks to take over the application, change stored procedures, steal or destroy data, and delete tables.
- Grant application DBAs access to only the application schema objects, not database-wide privileges, even if this is less convenient. Earlier in this chapter, schema privileges in Oracle AI Database 26ai were discussed as a feature that helps support this approach.
- Fully audit all activities for accountability and tracking.

Protecting data analysts or business intelligence users

These accounts allow access to data but, in most cases, should not permit data to be altered, deleted, or inserted.

- Avoid grants of system privileges such as SELECT ANY TABLE. Use direct object grants, either to the user or to a role, or use schema-level grants.
- Audit these users for misuse. Are they logging in from unusual IP addresses or running different programs than normal? These can be indicators of account compromise. If the users have static IP addresses, consider using SQL Firewall or Database Vault connect rules to restrict accounts to specific workstations. SQL Firewall and Database Vault are discussed later in the book.
- Audit these users for insert, update, and delete activity.
- Consider making these accounts read-only. Read-only accounts are a new feature of Oracle AI Database 26ai.

Protecting other database users

These accounts often require closer review because standard policies for common account types may not apply.

- Regularly confirm the account is still required, especially when assigned to a human actor.
- Track system privilege grants such as `SELECT ANY TABLE` and use object or schema privileges where possible to limit scope.
- Audit these accounts for misuse. Are they logging in from unusual IP addresses? Do patterns suggest credentials are shared by multiple people? Are they running different programs than they normally do?
- For accounts tied to batch processing where the same SQL statements are consistently issued, consider protecting them with SQL Firewall.
- Use Privilege Analysis to identify unnecessary privilege grants.

Agentic AI and MCP connections

Agentic AI and Model Context Protocol (MCP) connections typically fit into the same user categories as other nonhuman database access. In many implementations, they operate as application service accounts because they connect without a person directly initiating each database session and perform actions on behalf of application users, such as reading and writing application data or executing stored procedures. In other cases, an agent or MCP connection behaves more like an integration or automation component and is better treated as another database user, similar to a batch process or specialized tool account with narrowly defined access needs. Classify the account based on how it is used, then apply least privilege, enforce strong authentication, and audit all activity. These connections can be highly capable and, if overprivileged, significantly amplify the impact of a credential compromise.

With these access control patterns in place, the next step is to see how they work in practice. Hands-on exercises can help reinforce the concepts and provide a safe way to explore privilege design, user assessment, and entitlement monitoring.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Privilege Analysis
- Data Safe Fundamentals
- Oracle Database Security Central

Summary

Database access control is both a security requirement and an operational discipline. Compromised access can look like ordinary login activity, so every unnecessary privilege can expand risk. Effective programs keep authorization design at the center of database security, apply least privilege consistently, and use Oracle AI Database access control features with clear intent.

That work starts with understanding privilege scope. Oracle AI Database supports four privilege types—object, schema, system, and administrative—each with progressively broader scope and corresponding risk. The examples in this chapter show how scope changes risk: a targeted SELECT grant is limited to one table, a schema-level grant expands access within a schema, SELECT ANY TABLE extends access broadly, and SYSKM grants authority over encryption key management.

Roles provide a scalable way to manage privileges across users and teams. By grouping privileges into task-based and organizational roles, administrators can simplify onboarding, reduce repetitive grants, and keep access aligned with job responsibilities. To understand what a user can do, administrators should review role grants together with system, schema, and object grants. Oracle Data Safe User Assessment and Oracle Database Security Central provide practical ways to analyze user and role access.

Account type matters. DBAs, security administrators, application service accounts, application administrators, developers, testers, data analysts, and specialized tool accounts each carry different risk profiles. Access controls should reflect those differences. Use narrowly scoped privileges where possible, prefer schema privileges over broad system privileges when appropriate, monitor for unusual behavior such as unexpected IP addresses or client programs, and audit privileged activity for accountability.

Centralized identity and authorization can also reduce administrative overhead and help prevent orphaned accounts. Enterprise User Security integrates with Oracle directory services, Centrally Managed Users integrates with Microsoft Active Directory, and token-based access through OCI IAM or Microsoft Entra ID can map external identities and groups to database schemas and roles. Because Enterprise User Security is deprecated in Oracle AI Database 26ai, organizations using it should plan accordingly.

The next chapter, Separation of Duty, explains how Oracle Database Vault helps enforce stronger boundaries around powerful accounts. It shows how controls can restrict access to protected schemas and SQL commands, helping organizations separate operational administration from direct data access.

Chapter 7

Enforcing Separation of Duties

Why “who can do what” still breaks security

Many organizations invest heavily in perimeter defenses, identity systems, and monitoring. Yet damaging incidents can still occur through a simple path: a person with legitimate access does something they shouldn’t, whether intentionally or by mistake.

For databases, that risk often concentrates around powerful administrative accounts, shared operational responsibilities, and emergency break-glass practices that gradually become everyday workflows. The result is a recurring pattern: audit findings, control exceptions, and difficult questions about whether critical data is protected from insider misuse.

At the center of this challenge is a foundational security principle: Separation of Duty.

What separation of duty means and why it matters

Separation of duty (SoD) is the practice of splitting critical tasks and privileges across multiple individuals or roles so no single person can carry out a sensitive process end-to-end without oversight. Put simply, it helps ensure that the same person cannot both make and approve a change, or both administer the system and access the data without controls.

Compliance initiatives rarely fail because a control doesn't exist on paper; they fail because the control cannot be enforced consistently in day-to-day operations. Regulations and frameworks such as SOX, PCI DSS, HIPAA, GDPR-aligned governance programs, and common internal control standards all converge on the same expectation: privileged access must be limited, monitored, and constrained to job function.

SoD supports those expectations by reducing the following:

- The risk of fraud and unauthorized changes going undetected because collusion or independent review becomes necessary
- The blast radius of compromised administrator credentials because "admin" doesn't automatically mean "data access"
- Audit scope and remediation churn because roles and responsibilities become clearer and demonstrably enforced
- Operational fragility because teams stop relying on shared superuser access as the default way to get work done

What separation of duty is trying to prevent

Stealing sensitive data via compromised privileged-user accounts is one of the most common attack vectors for database breaches. Although database administrators may prefer a single account for convenience, dividing duties across multiple accounts and knowing exactly which privileges are in use can significantly improve security.

For example, the person who creates a new database account and sets its initial credentials does not need to be the same person who grants that account access to data. Separating account creation from data access prevents any one person from creating a new pathway to data, and requires two people working together to introduce that pathway. Similarly, in encryption key management, the person who can create and rotate encryption keys should not normally be the same person who manages the rest of the database, reducing the chance that both the database (or database backup) and the encryption keys are compromised at the same time.

SoD also reduces human error by increasing the likelihood that mistakes are caught and fixed before they cause harm.

Separation of duties supports compliance

In many environments, privileged accounts act as an exception mechanism that bypasses otherwise well-designed application controls. Even with a least-privilege model at the application layer, database superuser capabilities can undermine SoD goals when administrators can read sensitive tables, disable auditing, or alter security configurations during routine maintenance.

This often creates a gap between “policy intent” and “technical enforcement.” Organizations can describe their SoD model, but proving consistent enforcement across administrators, tools, scripts, and emergency procedures is difficult.

Separation of duties divides tasks among multiple users to reduce fraud, errors, and malicious activity, and to limit the damage from compromised accounts. Cybersecurity and regulatory concerns continue to drive strong controls for insider and privileged administrative accounts.

In the sections that follow, this chapter explores how capabilities in Oracle AI Database help organizations implement stronger separation of duty, reduce insider risk, and meet compliance expectations with less operational friction, without preventing DBAs from doing the work required to keep the business running.

Separating the duties

How far an organization can separate duties across different people often comes down to scale. If there are only one or two DBAs supporting the entire organization, the separation that can be applied in practice will be limited. In that reality, it is rarely feasible to staff distinct roles like a backup and recovery DBA or a performance tuning DBA.

In these cases, some organizations use a multiple account strategy. One person performs most day-to-day work with one account, and then uses a second account, with different privileges, for a small set of critical functions, such as security administration. The same person has credentials for both, but must log in with the account that matches the task at hand. This simple discipline can reduce human error and shrink the blast radius if an account is compromised, while still leaving the organization without strong protections against a malicious administrator.

The goal is to strike a practical balance: use a best-practice approach to separation of duties that lets privileged users do their jobs while helping protect sensitive data.

Enforce separation of administrative privileges

Most DBA work does not require sweeping, all-access permissions. In practice, the majority of routine operations can be handled with narrow, purpose-built privileges, allowing multiple administrators to manage the database without the risk that comes with using the all-powerful SYSDBA administrative privilege. `SYSOPER` is a clear example: it enables limited tasks such as starting and stopping the database, without handing over the full range of `SYSDBA` capabilities. The Oracle AI Database provides specialized administrative privileges, including `SYSBACKUP`, `SYSDG`, `SYSKM`, and `SYSRAC`, to support database backups, Oracle Data Guard administration, key management, and Oracle RAC management, respectively.

A practical way to reduce privilege abuse is to divide database administration into distinct administrative, operational, and security responsibilities, so no single administrator has unrestricted control over the database. The Oracle AI Database does allow all administrative privileges to be mapped to a single operating system (OS) group. However, that approach is not recommended because it does not enforce separation of duties. The stronger best practice is to create a separate OS group for each administrative privilege and associate each group with its corresponding privilege at the database binary level during installation.

For local connections, where an administrator first accesses the database host server and then logs in to the database, the database owner’s OS account (typically 'oracle') should not be used. Instead, administrators should use individual OS accounts that are members of the appropriate OS group or groups, which improves accountability and reduces

security risk. It is also a best practice to avoid using the client included with the database binary installation and instead install a separate client for administrator use.

Control the database owner account

The SYSDBA administrative privilege grants full access to the SYS (database owner) account and should be restricted to use during database upgrades and patching. Change management processes and privileged access management (PAM) systems should be used to control remote access to accounts with this privilege, including the default SYS account.

Practical tips for a secure database environment

A strong database security posture requires a few proven techniques. The following sections outline complementary controls that improve overall security. These common best practices are found consistently in environments where database security is paramount.

Implement a trusted path to data

Even if administrator credentials are compromised, access should be granted only when it comes through a trusted path. That trusted path can include checks such as IP address, the program used, time of day, authentication type, and other conditions. Adding these authorization checks makes it much harder for an attacker to succeed with stolen credentials. This is especially important for application service accounts, which are often prime targets.

Enforce least privilege

Grant database users only the minimum set of privileges required to perform their intended tasks or functions, and no more. Use privilege analysis to identify unnecessary privilege and role grants. When assigning privileges to a user or role, grant specific object or schema privileges rather than broad system privileges that enable access to all objects in the database. Similarly, create database roles with only the privileges needed for a particular function instead of using powerful roles such as the built-in DBA role. Granting one or more task-specific roles to a user enables a closer fit to the work the user needs to perform without adding unnecessary privileges. This least privilege model helps reduce the blast radius of a compromised account by limiting what an attacker can do within the database.

Do not share accounts

Sharing database accounts for convenience is common, but it comes with real cost. It removes accountability and increases risk because multiple people can use the same credentials. Each DBA should have an individual, named account with carefully tailored privileges and roles. If there are many DBAs or many databases, consider simplifying administration by centralizing database authentication and authorization in an external identity service such as Active Directory.

Safeguard the audit trail

Audit records provide a record of actions on a database, directory, or operating system. Activity by privileged administrators should be audited. Details such as the privileged user actions taken (`CREATE USER, CREATE ANY TABLE, ALTER SYSTEM, ALTER SESSION`), along with event context such as initiating IP address, event time, and the actual SQL statement, are examples of the audit information commonly needed for compliance and forensic reports. The `AUDIT_ADMIN` role grants privileges to change audit policies and purge audit records. This role should normally be granted only to the security DBA and to trusted tools or services used to collect and aggregate audit data. Further, if audit records can be created when administrative users access application accounts, alerts can be raised for further action.

Mandatory access controls with Oracle Database Vault

Oracle Database Vault is designed to help customers move from SoD as a guideline to SoD as an enforceable, testable control inside the database. Rather than relying solely on trust, process, or after-the-fact detective measures, Database Vault helps establish durable boundaries around sensitive data and administrative actions, ensuring that even highly privileged users operate within defined rules aligned to role and business need.

Database Vault is one of the most powerful security options for Oracle AI Database 26ai. Database Vault is a multi-purpose access control mechanism that addresses many different use cases. Primary use cases for Database Vault are as follows:

- **Blocking privileged user access to data**: Database Vault prevents privileged users, such as DBAs, from accessing sensitive application data. This helps meet regulatory requirements and reduce risk. Even if DBA credentials are compromised, sensitive data remains protected, minimizing the potential for data breaches.
- **Enforcing a trusted path**: Database Vault helps ensure that application service accounts, often targeted by attackers, can only connect to the database under strict conditions. Rules can restrict these accounts to specific IP addresses, programs, or operating system users.
- **Controlling SQL command execution**: Database Vault allows precise control over database commands, enabling conditions like requiring multiple administrators for critical actions (e.g., no table can be dropped unless two DBAs are logged in) or restricting user actions to specific times or circumstances.
- **Controlling administrator access**: In most cases, there is no reason for database administrators to have access to the application schemas and data. Using the different administrative privileges does not prevent the administrators from accessing user-specific data. Read on to see how to enforce separation of duties further and break the link between database administration and data access.

The out-of-the-box DBA role is granted most system privileges, including the powerful `SELECT ANY TABLE` privilege. With this privilege, a DBA can view any data in the database, including salary, taxpayer IDs, phone numbers, corporate financial forecasts, intellectual property, and other sensitive data. If a cybercriminal successfully compromises the credentials of a DBA, they can now easily access and exfiltrate sensitive data.

Other privileged accounts may also be granted `SELECT ANY TABLE`, frequently by administrators who are trying to quickly solve problems related to their access. Complex applications with multiple schemas frequently use system privileges instead of object privileges, making it possible for exploits like SQL injection attacks to access data throughout the database. Like DBAs, these accounts create an elevated risk to data and are prime targets for attackers.

Oracle Database Vault can restrict privileged users, including database administrators, from accessing sensitive data with an access control mechanism called a *security realm*. Security realms are collections of schemas or specific sensitive objects where access is controlled by Database Vault. Database Vault policies override system privileges like `SELECT ANY TABLE` for objects and schemas protected by a security realm. With Database Vault, database administrators can manage the database but cannot access sensitive schemas/objects protected by realms.

Figure 7-1 shows an example of a realm protecting sensitive data within the human resources (HR) schema. The Database Vault realm prevents users with the powerful DBA role from accessing data inside the HR realm yet still allows them to do their database administration tasks like creating indexes. Business users with a valid requirement to view data are still able to access data because they have permissions within the realm.

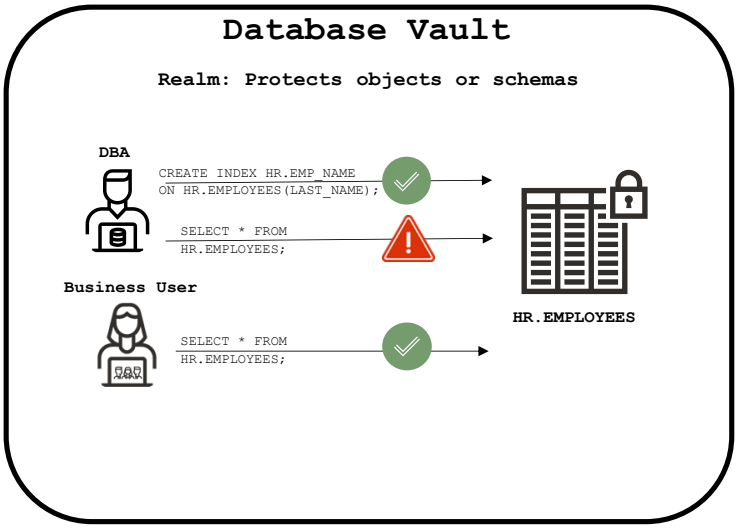


Figure 7-1: Database Vault Realms block administrator access to data

Enforcing a trusted path

Access within a realm is controlled by Database Vault rule sets. A rule set is one or more programmatically evaluated rules that define the conditions for access. In the example shown in Figure 7-2, the rule set includes two rules: one checks the source IP address of the user connection, and the other checks that the time-of-day is within approved business hours. The business user must satisfy both rules to access data in the realm. By enforcing these rules, Database Vault shrinks the attack surface and enables a trusted path to data through multi-factor authorization policies, further reducing the risk from compromised accounts.

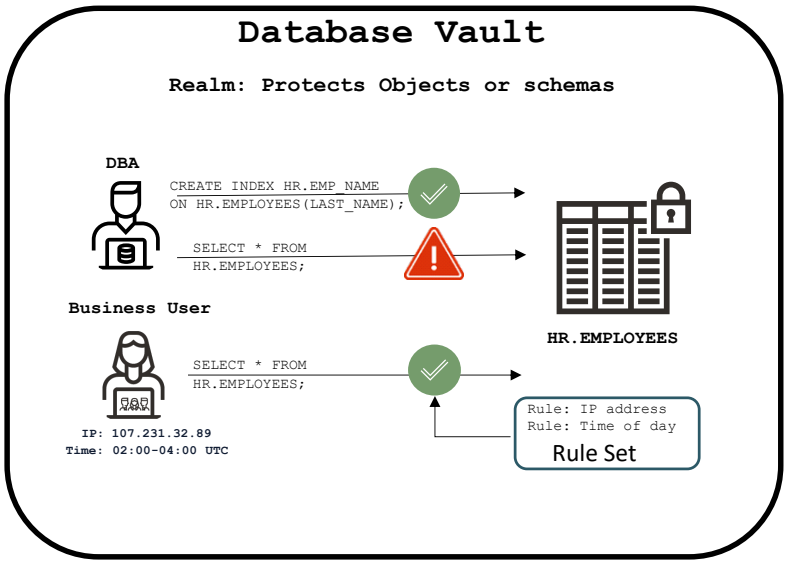


Figure 7-2: Database Vault uses rules to make access control decisions

A database can contain many different Database Vault realms, each with its own set of authorized users controlled by rule sets appropriate to those users and the realm. In Figure 7-3, we can see that the human resources application administrator can manage the HR data but cannot access objects or data within the FIN realm, even though the HR application administrator has broad system privileges. Similarly, the highly privileged finance application administrator can view data and objects within the FIN realm but not within the HR realm. Thus, Database Vault enforces separation of duties at the database object and schema level, even overriding system privileges, and isolates the DBA from application data.

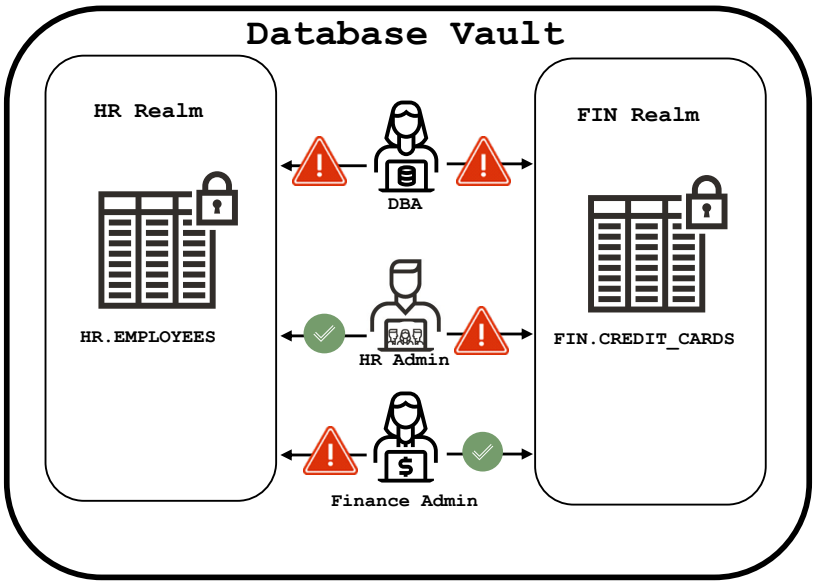


Figure 7-3: Oracle AI Database can have multiple realms in the same database

By default, realms are overridden by schema ownership and direct object grants. In Figure 7-3 above, that would mean that the HR account (the schema owner of the EMPLOYEES table) would be able to access data in the EMPLOYEES table and the FIN account (owner of the CREDIT_CARDS table) would be able to see data in the CREDIT_CARDS table even though they were not authorized within the realms. Sometimes this is the desired behavior, but there are situations where the realm needs to control ALL users, regardless of ownership or direct object privileges. Having the realm override object privileges can be valuable when an administrator must make changes during an application upgrade or patching. In some upgrades, only application procedures and functions need to be modified, not the tables and views that store sensitive data. In that case, stronger protection can be applied to sensitive tables and views while still allowing access to update the procedures.

For this use case, Database Vault realms can be made “mandatory,” meaning not even the schema owner can access objects protected by the realm unless access is explicitly granted through Database Vault. Mandatory realms override both schema ownership and direct object grants. With a mandatory realm, users are not allowed access, even to objects they own, unless they are specifically granted membership to the realm through Database Vault.

With Database Vault, the only user who can grant access to protected data is the security administrator, not the data owner. This simplifies proving who has access because auditors do not need to trace a potentially complex chain of direct object grants across users and roles. Instead, permissions to access or manipulate objects or schemas are maintained in clear, easy-to-understand security realms.

Separation of duties within Database Vault

As you might expect, separation of duties is built into Database Vault. Enabling Database Vault creates additional roles intended to enforce the separation of duties. These roles include:

- **Database Vault Account Manager** provides out-of-the-box enforced separation of duties that controls the creation and management of users in the database along with their passwords. By default, when Database Vault is enabled, the DBA role loses the ability to create users or change their passwords.
- **Database Vault Administrator and Owner** roles allow a security administrator to create, modify, and manage Database Vault security controls, including the ability to manage realms and add realm participants. The Database Vault Administrator role controls the various PL/SQL packages associated with Database Vault. The Database Vault Owner role includes the Database Vault Administrator role, but also allows a user with the role to manage and monitor the Database Vault configuration.

This split between database administration and user administration helps block a common attack pattern: an attacker steals DBA credentials, then uses that legitimate access to create a rogue user and grant powerful privileges, making it easier to steal sensitive data. That rogue account can also serve as a convenient way back into the database if the DBA credentials are later changed.

Because of this, Database Vault role membership needs careful control. If an organization cannot support a separate security administrator, and SoD must be relaxed, the Database Vault roles can be granted to the DBA role so DBAs can manage both users and Database Vault policies. The tradeoff is important: granting these roles back to the DBA role also brings back the risk that a compromise of a database administrator account could result in a compromise of sensitive data.

The database security assessment tools report which users hold Database Vault roles. Oracle Data Safe and Oracle Database Security Central also detect drift in Database Vault role assignments, making changes straightforward to spot.

Oracle Database Vault can also separate who can view and manage audit records from privileged users such as database administrators. This additional separation matters because purging audit records is a common way to hide malicious or inappropriate activity. Oracle AI Database 26ai adds Database Vault control over the `AUDIT_ADMIN` and `AUDIT_VIEWER` roles.

The database administrator team cannot access application data unless they are explicitly granted permission within the realm that protects the data, but they can still perform common DBA tasks like performance tuning, memory management, backup, and others. Certain DBA tasks like exporting data or scheduling automated jobs may expose sensitive data to the DBA and will require additional authorization by the Database Vault security administrator.

Separation of duties with containerized databases

Oracle introduced the concept of container databases in Oracle Database 12c. In a containerized environment, a container root database (CDB) stores common metadata and settings that apply across the container databases. A pluggable database (PDB) is a collection of schemas, schema objects, and non-schema objects that appears to an application or user as a separate database. A single CDB can host many PDBs. The container architecture offers numerous benefits, including simpler maintenance, portability across servers, and increased resource utilization with lower overhead.

From a security standpoint, the container architecture adds value by providing another way to separate data and users into different pluggable databases. Each pluggable database can maintain its own policies, administrators, and configuration.

Starting with Oracle Database 19c, Database Vault's Operations Control feature can block common users, such as infrastructure DBAs, from accessing application data in pluggable databases. Using Database Vault Operations

Control to separate common users from local data in PDBs does not require creating separate security realms. As organizations adopt Oracle AI Database 26ai, which always uses container databases, Operations Control becomes more important than ever.

With Operations Control, the Database Vault administrator can do the following:

- Perform all administrative activities on the container database.
- Perform all administrative activities on the pluggable database, except accessing or managing local user objects or local user data.
- Grant a common administrator an exception to access PDB local data or exempt a PDB from Operations Control enforcement.

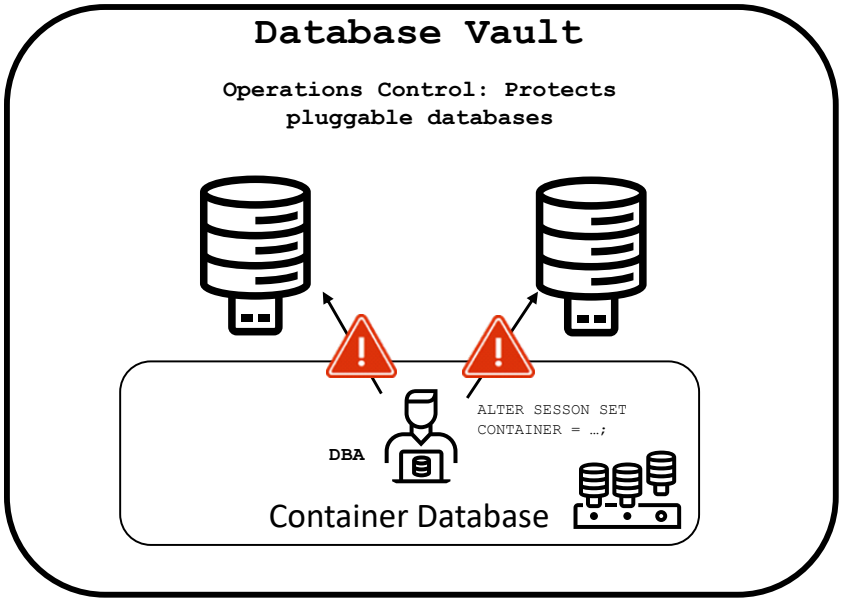


Figure 7-4: Operations Control protects pluggable databases

Control SQL database commands with Oracle Database Vault

This topic is not strictly part of separation of duties, but it fits naturally in a discussion of Database Vault. Many organizations have change management policies that prohibit dropping tables, modifying stored procedures or other database objects, and changing system parameters outside defined maintenance windows. Database Vault can enforce these change management policies, reducing the risk of accidental or malicious violations. This matters because human error is a leading cause of production outage, and DBAs often work across dev, test, and production terminal sessions where mistakes are easy to make.

Database Vault provides fine-grained SQL command rules to help prevent accidental or malicious activity. These command rules can either prohibit specific commands or use rule sets to define the conditions under which commands are permitted. As with realms, the rule sets attached to a command rule can be based on anything that can be checked programmatically. For example, changes to production schemas, such as dropping a table, can be prohibited unless two application administrators are logged in at the same time, the person executing the change connects from an approved jump server, the work occurs outside normal business hours, and a change number is set for the session.

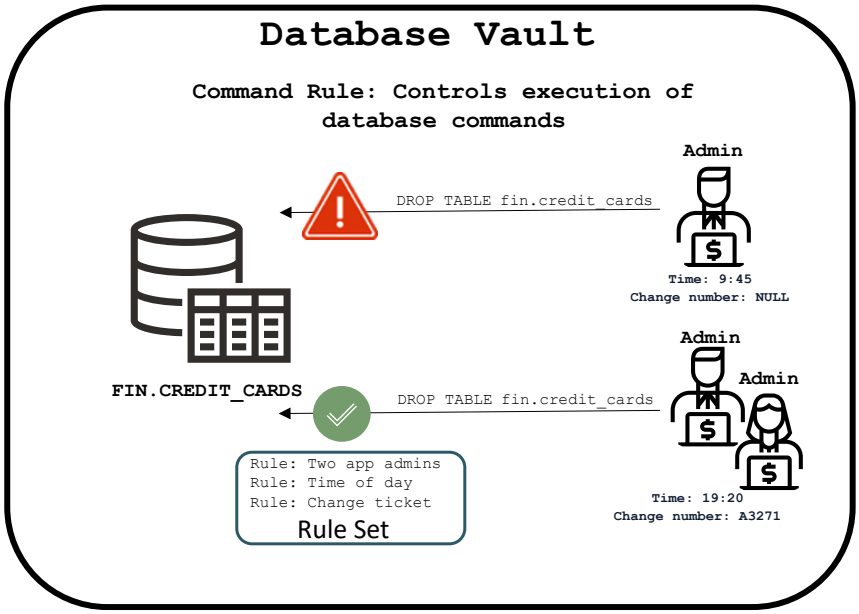


Figure 7-5: Example Database Vault Command Rules

As another example, DBAs can be limited to running a `DROP TABLE` command only from their work desktops and only during working hours, reducing the risk of unauthorized remote access after-hours. Other policies might restrict SQL commands such as `CREATE`, `TRUNCATE`, or `ALTER TABLE` to maintenance windows. Database Vault can also enforce conditions like requiring that changes must be associated with a trouble ticket number or ensuring that two DBAs have logged in at the same time before a particularly risky activity (like rekeying the database) can succeed.

Break-glass scenarios

As a general principle, operating system administrators, database administrators, and developers should have limited or no access to production data. However, access is sometimes necessary, such as when performing application upgrades or troubleshooting application schema or application data issues.

When these scenarios arise, administrative and development staff should have a defined mechanism to gain access to the production schema and production data. This is commonly referred to as a break-glass scenario. When an administrator breaks the glass, they temporarily acquire elevated privileges to complete a specific task. This can be

done by checking out a privileged account and using it to perform the required actions, or by enabling an Oracle AI Database role that grants a day-to-day account a higher level of database privileges and access to production data.

In the first approach, where an account is checked out, a highly privileged user account can be stored in a Privileged Access Management (PAM) solution. The person needing access checks out the account, completes the task, and then checks the account back into the PAM solution. In this scenario, Database Vault can be used to constrain where and how the checked-out account can be used. For example, the account might be permitted to connect only from a specific subnet or a specific trusted host. The checked-out account might be allowed to run `CREATE` or `ALTER` commands but not `DROP TABLE`, `TRUNCATE TABLE`, or `DELETE` commands on production schemas. This allows the account to view data while restricting or blocking destructive actions. Once the emergency is resolved, the checked-out database account can be automatically locked by the PAM solution.

In the latter scenario, the person needing access can enable a database role called a secure application role. Secure application roles are not granted directly to a user. A secure application role can be enabled only through its associated PL/SQL procedure or Database Vault rule set, depending on the implementation (read more about secure application roles).

In this approach, the role can be enabled only if the requesting database user meets the criteria defined in a Database Vault rule set. For example, the rule set might allow the role to be enabled only when the user connects from trusted hosts, IP addresses, or subnets. The rule set can also restrict enablement to specific hours or specific days. Access can be further limited to a defined list of database users, such as only senior database administrators.

When the emergency is resolved, the database user logs off and the role is automatically revoked until it is needed again. This model is similar to using sudo on Linux systems or “open as Administrator” on Windows systems: the privileges are available, but they are not continuously enabled.

In day-to-day operations, each administrator should use a least-privilege, named account with tightened domain-specific roles. Break-glass privileges are specifically for unplanned crises and controlling them behind robust administrative policies helps maintain a secure database environment. This way, if an attacker compromises ordinary credentials, they still face the barriers of multiple approval levels, rotating credentials, and the blend of preventive and detective controls that enable a defense-in-depth approach.

Operationalizing Oracle Database Vault

There are three key elements to consider when operationalizing Oracle Database Vault:

- What needs to be protected, and under what conditions.
- Which SQL commands should be allowed in your database environment.
- What process and organizational changes are needed because of the newly enforced separation of duties.

Before defining Oracle Database Vault realms, it is important to first identify what data should be treated as sensitive. Some organizations take a broad stance and assume all application data is sensitive, creating a realm for every application schema. Others prefer a more targeted approach and use sensitive data discovery to pinpoint exactly what needs protection. Either path can work, and Database Vault can support both.

To decide which SQL commands should be allowed, many teams start with audit records. Audit trails can show which critical SQL commands, such as `ALTER SYSTEM`, `ALTER DIRECTORY`, and related actions, were executed, along with the context around them, including IP address, program name, database, or operating system username. Database Vault realms and SQL command rules can then be used to constrain those commands by factors like time of day, source IP address, and other criteria, helping ensure changes align with policy.

At the same time, special attention is needed for separation of duties inside Database Vault, since Database Vault is the mechanism used to enforce separation of duties across the rest of the database. Many organizations roll this out in stages. For example, Database Vault roles might be granted to the DBA role at first and then separated later as

processes mature. This approach can accelerate early control adoption while spreading SoD change over a longer time frame.

Some database teams hesitate to enable preventive controls like Database Vault because they worry about outages or breaking administrative scripts. To help reduce that risk, Database Vault offers a simulation mode. In simulation mode, realms and SQL command rules are evaluated, but instead of blocking activity, Database Vault logs policy violations. This enables full end-to-end regression testing without disrupting application operations. After testing confirms that legitimate activity would not have been blocked, policies can be moved from simulation mode to blocking mode.

Finally, violations need to be captured and alerted. These audit records are especially valuable because Database Vault alerts can signal either early probing by a malicious user or a need for additional training on how to access sensitive data appropriately. Oracle Data Safe and Oracle Database Security Central can both be used to collect and report on Database Vault violations.

Database Vault for vibe-coded apps and privileged MCP database access

Vibe-coded applications, built quickly through iterative experimentation, often inherit risky defaults, including reused credentials, overprivileged database connections, and minimal security hardening. What begins as a rapid prototype can become a direct pathway to sensitive data. A single SQL injection flaw, leaked credential, or coding error can expose entire databases. Opening an MCP connection with a privileged user further amplifies risk by introducing a new, automatable pathway that can rapidly enumerate objects, run high-volume queries, and exfiltrate sensitive data. Returned results may also be captured in agent and tool logs outside approved data handling controls.

Oracle Database Vault mitigates these risks by enforcing policy at the database layer, independent of application code quality. With Database Vault realms, sensitive schemas and tables, such as PII, finance, and HR, can be protected while also overriding powerful system privileges for example, blocking access even when an account has `SELECT ANY TABLE`.

With mandatory realms, explicit authorization can be required even for schema owners, preventing “god accounts” and service accounts from accessing protected data by default. With trusted path rule sets, privileged service accounts, including MCP accounts, can be restricted by where they connect from, such as approved subnets or hosts, time windows, and client attributes.

With command rules, high-risk SQL, including DDL, privilege changes, and destructive operations, can be prohibited or tightly constrained so compromised privileged credentials cannot be used to alter security posture or damage data. Together, these controls enforce separation of duties and can control, or completely forbid, privileged access to sensitive data from AI-generated prototype applications and privileged Model Context Protocol (MCP) database access.

Database Vault and packaged applications

Several Oracle and third-party applications are certified to work with Oracle Database Vault, including Oracle Fusion Cloud Human Capital Management, Oracle Fusion Cloud ERP, Oracle E-Business Suite, Oracle PeopleSoft, and SAP. Review the certification matrix on Oracle Support for more information about application certifications and the Oracle Cloud website for cloud support.

Administration of Database Vault using Oracle Database Security Central

Database Security Central can help manage, simulate and view Database Vault policies on your Oracle databases. Unified policy management in Database Security Central helps enforce consistent controls across the fleet. The controls include audit, alerting, Database Vault, Database Firewall, and SQL Firewall policies. Figure 7-6 shows the Database Vault policies console, which enables users to see the Database Vault enforcement across their fleet, along with violations generated for non-conforming access patterns.

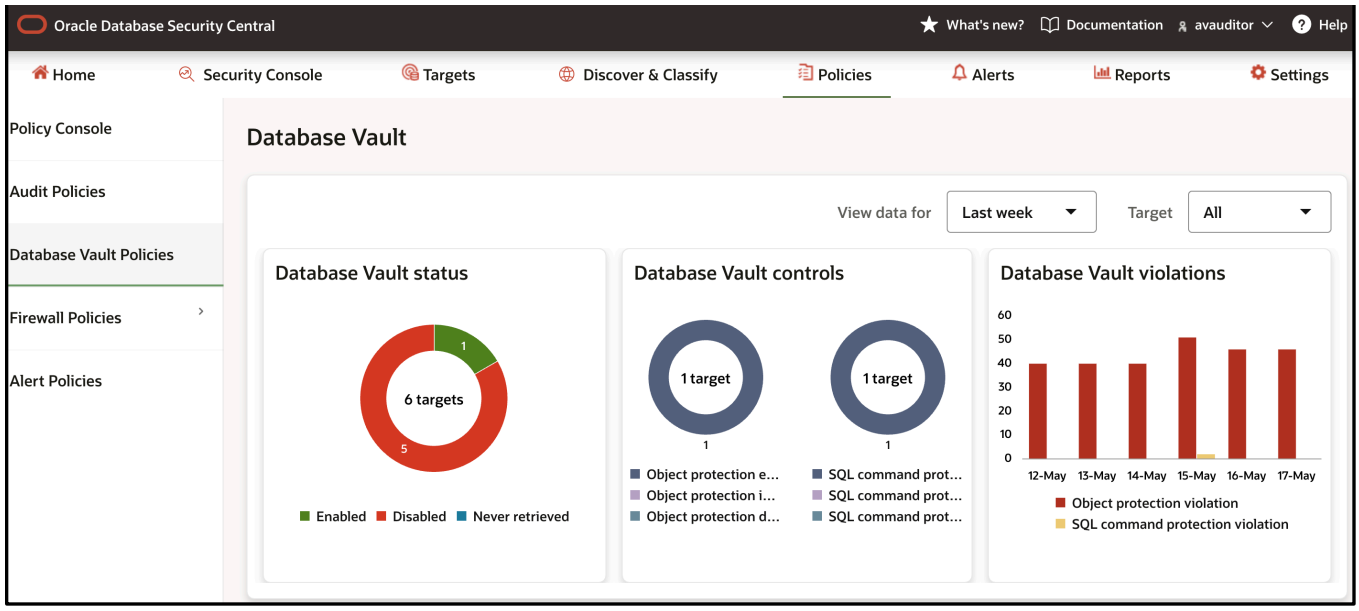


Figure 7-6: Database Vault policies console in Database Security Central

Figure 7-7 outlines the Database Vault object protection and SQL command protection configured for one of the Oracle Database instances. Object protection policies represent the Database Vault realm protection. SQL command protection policies represent the Database Vault command rules. Administrative controls including auditing (available as of Oracle AI Database 26ai and later), Data Pump operations, Scheduler jobs, Proxy authentication, and Data Definition Language (DDL) operations, can be managed directly in Security Central.

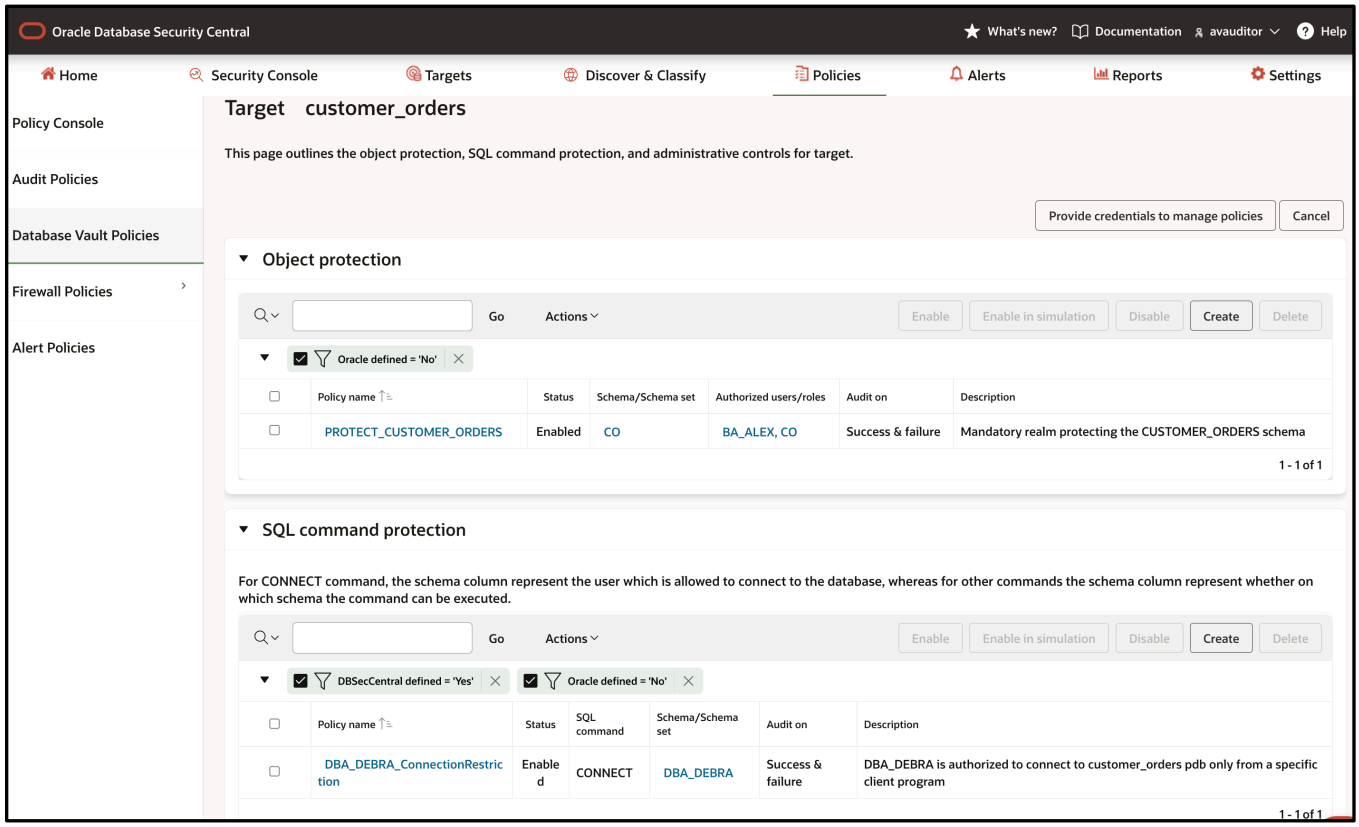


Figure 7-7: Example Database Vault policy configuration in Database Security Central

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- DB Security- Database Vault

Summary

Separation of duties (SoD) is framed as a practical response to a persistent database security reality: many of the biggest incidents don't start with "unauthorized access." They start with legitimate access being misused, especially when powerful administrative accounts and break-glass procedures quietly drift into everyday habits. SoD tackles this by splitting critical tasks and privileges so no single person can run a sensitive process end-to-end without oversight, cutting directly into insider risk and privileged access exposure.

From a compliance perspective, the message stays simple and grounded. Programs rarely fail because a policy doesn't exist. They fail when enforcement erodes in day-to-day operations. SoD supports regulatory expectations to limit and monitor privileged access, while also reducing fraud exposure, containing the impact of compromised credentials, and minimizing audit and remediation churn.

The principle becomes workable through patterns that fit real-world constraints, including smaller teams that can't cleanly separate every role. Practical options include using multiple named accounts, moving away from broad superuser habits toward narrower administrative privileges, and adding OS-level and process controls that improve accountability. Reinforcing best practices like least privilege, trusted path requirements, avoiding shared accounts, and protecting audit administration further reduces the odds that privileged users can bypass controls or erase evidence.

Oracle Database Vault is positioned as the mechanism that makes SoD enforceable and testable inside the database. Realms and rule sets can override broad system privileges, trusted path controls can limit how privileged access is exercised, and SQL command rules can backstop change management policies. Rollout guidance keeps expectations realistic: adopt in stages, use simulation mode to log violations before blocking, use container database Operations Control to isolate pluggable databases, and govern break-glass access with constrained privileged accounts or secure application roles.

Database Security Central can help manage and view Database Vault policies on your Oracle databases.

The next chapter, Mitigating Risks from SQL Injection, moves from who can touch the data to how attackers slip through the smallest cracks in query logic and privileged pathways, turning convenience into catastrophe. Stay with the narrative, because the discussion is poised to shift from guarding roles and controls to confronting the fast, sharp mechanics of exploitation, where a single flaw can suddenly illuminate every table in the room.

Chapter 8

Minimizing Risk from SQL Injection

The persistent threat of SQL injection

SQL injection remains one of the most common and damaging routes attackers use to access data. Even after decades of attention, it continues to show up in data-driven applications with surprising regularity. The Open Worldwide Application Security Project (OWASP) has consistently ranked SQL injection among the top ten web application vulnerabilities and has repeatedly placed it in the top positions since its first list in 2004.

This staying power deserves a closer look. SQL injection persists largely because it is carried out through applications, which, by design, must be accessible to users and, in many cases, to the entire internet. Applications act as the public-facing gateway to data, and attackers take full advantage of that necessary exposure.

In theory, this vulnerability should be a thing of the past. Developers have been trained on SQL injection risks for years, and automated code scanning tools routinely flag potential injection points during development. Yet despite this training and tooling, SQL injection remains a persistent threat to data-driven web applications. The gap between what is known and what is consistently done points to an uncomfortable truth: producing perfectly secure code across large, complex applications every time is extraordinarily difficult.

Why SQL injection demands your attention

SQL injection is rarely just an isolated data access issue. More often, it becomes the opening move in a broader compromise. Attackers use SQL injection vulnerabilities to steal sensitive information, corrupt critical data, destroy database structures, and then pivot laterally from the database into other systems across the network.

The blast radius of a successful SQL injection attack can be significant. In many multi-tier architectures, applications connect to databases through powerful service accounts that have broad schema access and procedure execution rights. When an attacker exploits a SQL injection flaw, malicious queries run with the full privileges of that application account, which can put an entire data estate at risk.

This chapter explores two complementary ways to reduce SQL injection risk at or near the database layer: the network-based Database Firewall in Oracle Database Security Central (Security Central), and the database-resident SQL Firewall built into Oracle AI Database 26ai. It compares how these technologies work, outlines when each approach is the better fit, and provides decision criteria to support selecting the option that best aligns with the environment and requirements.

Understanding the SQL injection attack pattern

SQL injection is fundamentally an application layer vulnerability, not a database vulnerability. This distinction matters because it makes clear where ownership sits and where defenses need to be focused. The weakness appears when an application treats untrusted input as executable SQL, giving an attacker a way to inject and run their own statements through the application to read, modify, insert, or delete data and, in the worst cases, damage or destroy database objects and data.

SQL injection attacks come in many variations, including blind injection, time-based injection, union-based injection, and more. However, they typically follow the same underlying attack pattern, as illustrated in Figure 8-1.

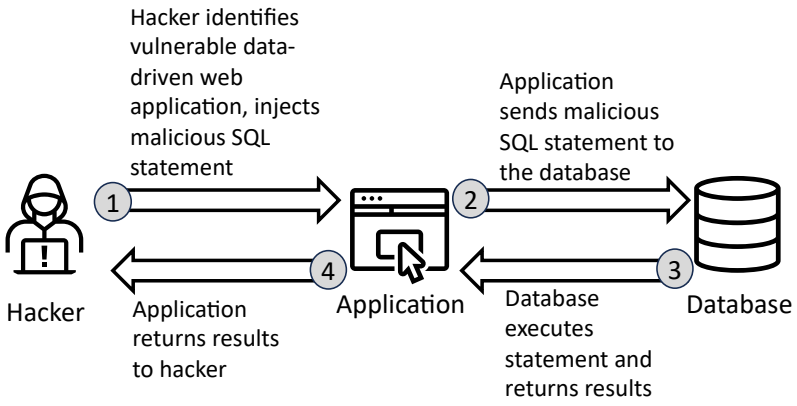


Figure 8-1: Example outlining SQL injection attack patterns

In multi-tier architectures, applications usually enforce access control in the middle tier, using the authenticated user's identity and authorization level to decide what they can do. SQL injection attacks sidestep these controls by slipping malicious SQL into input fields or URL parameters in data-driven applications. Because the injected statements run with the application account's privileges, an attacker can effectively bypass web application authentication and authorization altogether and gain direct ability to read, change, or delete sensitive data, including customer information, personal health records, trade secrets, intellectual property, and financial data.

This built-in privilege escalation makes SQL injection especially dangerous. A low privilege attacker who successfully exploits SQL injection can inherit the full privileges of the application database account, which often includes access to the entire schema and all stored procedures.

A defense in-depth strategy against SQL injection

Preventing SQL injection calls for a defense in-depth strategy that protects both upstream and downstream of the application tier. It can be viewed as building concentric rings of protection: each layer is designed to catch what the previous layer missed, steadily shrinking the attack surface.

Upstream protection (closest to the application user) typically includes deploying a web application firewall (WAF) to inspect incoming HTTP traffic for known SQL injection signatures. A WAF can block many common attack attempts before they ever reach application code. That said, signature-based tools come with built-in limits. They can miss novel attack patterns, zero-day exploits, and more sophisticated evasion techniques. A WAF reviews HTTP payloads, but it cannot assess the actual SQL with full database context.

Downstream protection focuses on the SQL the application actually issues, identifying and blocking malicious commands that make it past edge controls. This layer helps close gaps and reduces the blast radius when application vulnerabilities are present. This is where database firewalls deliver critical value.

As shown in Figure 8-2: Effective approaches to combating SQL Injection, the strongest strategy uses both approaches: block as much as possible at the edge with a WAF, while also protecting the database to contain risk when application vulnerabilities exist.

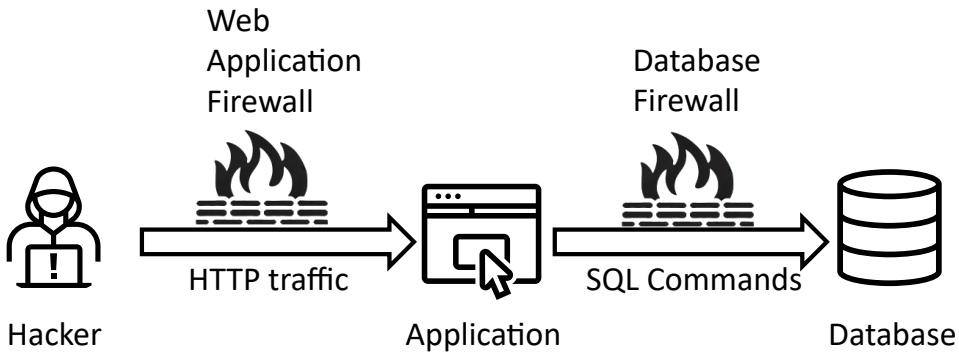


Figure 8-2: Effective approaches to combating SQL Injection attacks include the use of multiple firewalls

Database firewall solutions from Oracle

Oracle Database security offers two distinct approaches to mitigate SQL injection risk for data-driven applications:

- **Network-based Database Firewall** in Security Central
- **Database-resident SQL Firewall** operating inside the database kernel (introduced in Oracle AI Database 26ai)

Unlike web application firewalls that rely on regular expressions and signature matching, these database-centric approaches treat SQL injection prevention as an access control problem. Each solution learns your application's typical SQL patterns during normal operation, builds an allowlist of permitted statements, and rejects or alerts on anything that deviates from that established profile.

Security Central applies this protection at the network layer, positioning itself between the application and the database. SQL Firewall brings the same control directly into the database kernel, eliminating the complexity of routing database traffic through a separate appliance. Both options deliver powerful protection; the choice between them depends on your specific environment, requirements, and constraints.

Both network-based and database-resident firewalls address these core use cases:

- Provide near-real-time protection by restricting execution to authorized SQL statements arriving from trusted connection paths
- Reduce risk from SQL injection attacks, anomalous access patterns, and credential theft or abuse
- Enforce trusted database connection paths by blocking unauthorized connection sources.

Network-based Database Firewall in Security Central

Database Firewall, a core component of Security Central, acts as the database's first line of defense at the network layer. It continuously monitors incoming SQL traffic and enforces expected database behavior to help prevent SQL injection, application bypass, and other malicious activity. From a centralized deployment, a single Database Firewall instance can protect multiple heterogeneous databases across an environment, including Oracle Database, MySQL, Microsoft SQL Server, SAP Sybase, and IBM Db2.

As illustrated in Figure 8-3: Example showing how Oracle Database Firewall blocks SQL injection attempts, an attacker may be able to slip malicious input into an application field that the application fails to block, but Database Firewall can still stop the attack. If the SQL generated by the application does not match the allowlist of permitted statements, Database Firewall blocks the request before it reaches the database. It then sends detailed information about the attempted violation to the Audit Vault Server, which generates alerts and provides a centralized platform for analysis and reporting.

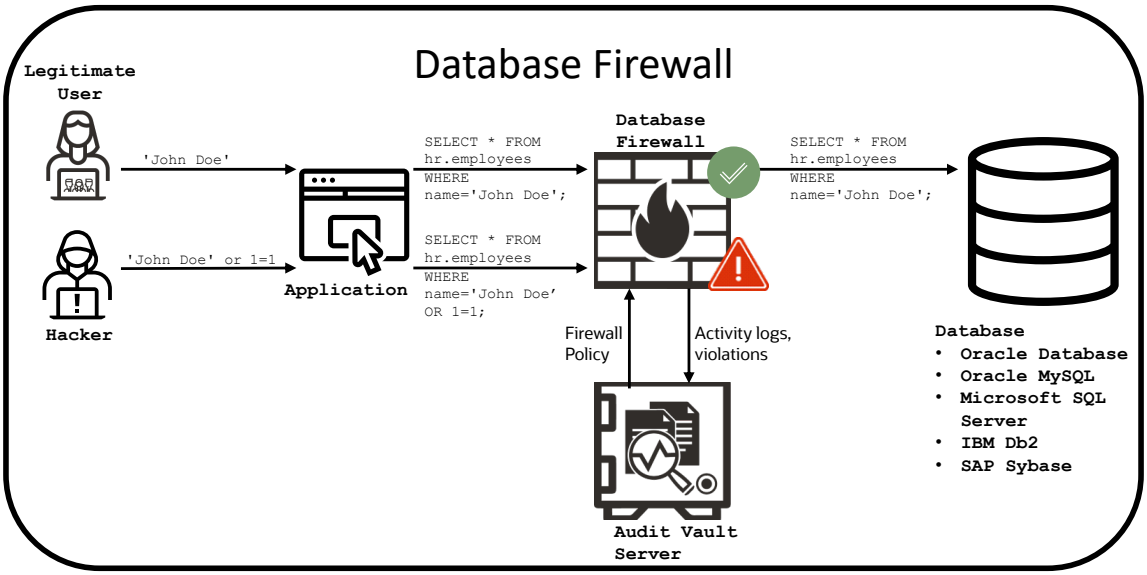


Figure 8-3: Example showing how Oracle Database Firewall blocks SQL injection attempts

Deployment modes for Database Firewall

Database Firewall supports three distinct deployment modes, each offering different capabilities and deployment characteristics. Table 8-1 summarizes these options.

MODE	Description	Monitoring or blocking
Proxy	All traffic to the database server is routed through the Database Firewall, including return traffic.	Monitoring and blocking
Host Monitor	An agent deployed on the database server captures SQL traffic destined for the database and securely forwards it for analysis.	Monitoring only
Out-of-band	Database Firewall listens to the network traffic sent to the database. Several technologies, such as SPAN ports, port replicators, and network taps, can send a copy of the database traffic to the Database Firewall.	Monitoring only

Table 8-1: Deployment modes of Database Firewall

Choose the deployment mode based on whether the goal is detection only or detection and blocking. If the priority is visibility without enforcement, any of the three modes can monitor traffic and surface policy violations. All modes work by sniffing network traffic and evaluating SQL statements against Database Firewall policies.

If the requirement is to actively stop malicious activity, Database Firewall must be deployed inline using proxy mode:

- **Proxy mode**: Database traffic flows through the Database Firewall, which inspects SQL in real time as it passes through. This is the only mode that can block unauthorized SQL. Proxy mode can be configured initially for monitoring only, generating alerts on violations while the allowlist is refined. After the allowlist

accuracy has been validated, it can be switched to blocking mode to prevent SQL injection and other unauthorized SQL from reaching the database.

- **Host monitor**: An agent installed on the database server captures incoming SQL and relays it to the Database Firewall appliance for analysis. This mode provides comprehensive monitoring and alerting without requiring changes to network paths or client configurations.
- **Out-of-band mode**: Database Firewall inspects a copy of network traffic from the database network segment. In most environments, a network switch or router mirrors traffic to the firewall using SPAN ports, mirror port, or network tap. This mode provides broad monitoring coverage without placing the firewall in the data path, helping minimize impact on database performance and availability.

How Database Firewall evaluates SQL traffic

Database Firewall inspects SQL traffic flowing into the database and decides whether to allow, log, alert, substitute, or block each statement. It evaluates traffic through multiple inspection stages, looking at the source IP address, database username, operating system username, client program name, SQL statement category (DDL or DML), and the database objects being accessed. This layered review determines whether a statement should be logged, trigger an alert, proceed to the database, or be blocked.

Database Firewall supports both allowlist and denylist policies, but allowlists generally deliver stronger protection. The universe of potentially malicious SQL patterns is effectively unlimited and constantly changing, while the set of statements an application legitimately runs is finite and usually well understood. Put simply, it is easier and safer to define what should pass than to predict everything that should not.

Managing Database Firewall policies

Database Firewall collects violation logs and activity records and forwards them to the Audit Vault Server for centralized analysis, alerting, and reporting.

Database Firewall policies are managed centrally through the Security Central console, with no separate Database Firewall management console. For comprehensive information on Database Firewall policy configuration, refer to the *Oracle Database Security Central Auditor's Guide*. The technical paper "Mitigating Risks from SQL Injection" provides additional practical guidance.

Database-resident SQL Firewall in Oracle AI Database 26ai

SQL Firewall is a new capability in Oracle AI Database 26ai that helps reduce SQL injection risk through kernel-level inspection and enforcement. It is available by licensing either Oracle Database Vault or Security Central, with no need to license both. SQL Firewall is included with either license.

Like Database Firewall, SQL Firewall mitigates risk from web application attacks such as SQL injection against data-driven applications. The key difference is architectural: SQL Firewall runs inside the database kernel rather than as a separate network appliance. Because it operates within Oracle Database, close to where the data resides, it sharply reduces bypass risk. SQL Firewall inspects every incoming SQL statement, whether local or over the network, encrypted or cleartext, and helps ensure the database executes only SQL that has been explicitly authorized through trusted connection paths.

SQL Firewall also uses an allowlist model. Running inside the database allows it to evaluate both session context and internal database metadata. With this richer view of allowlist rules, metadata, and session context, SQL Firewall can stop more sophisticated evasion techniques, such as encoded SQL, references to synonyms, or dynamically generated object names that may look different but resolve to the same underlying objects.

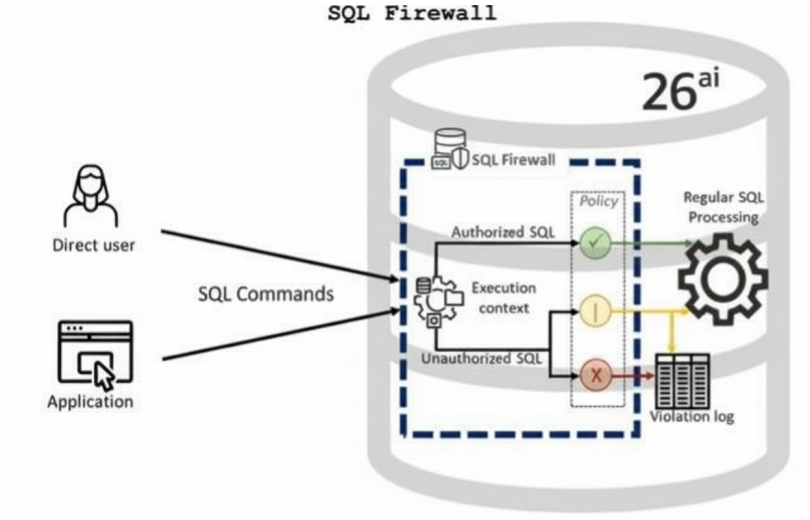


Figure 8-4: Oracle SQL Firewall is built into the Oracle AI Database 26ai kernel

SQL Firewall can monitor or block unusual access patterns, helping reduce risk without adding significant overhead. For each database connection, it validates the source IP address, operating system username, and client program name. If an account connects from a previously unseen IP address or uses an unexpected client program, SQL Firewall can flag or block the session based on the configured policy.

SQL command rules typically need longer training periods to capture the full set of permitted statements across workloads. Session context rules, by contrast, are usually quick to establish and require minimal training because they focus on connection attributes rather than SQL variations. Session context rules can also be defined manually when the environment calls for explicit control.

Unlike Database Firewall, which is heterogeneous and supports many database platforms and versions, SQL Firewall is built into Oracle Database and works only with Oracle AI Database 26ai and later versions.

Reporting on SQL Firewall violations

SQL Firewall records violations in the `DBA_SQL_FIREWALL_VIOLATIONS` view using Oracle Database fast-ingest architecture to help minimize performance impact. If needed, violations can also be audited, protecting the records with the audit trail's tamper-evident controls.

In multi-database environments, violation logs should be collected for consolidated analysis, alerting, and reporting. Oracle Data Safe and Oracle Database Security Central both provide centralized platforms for this purpose.

Administering SQL Firewall

SQL Firewall can be administered through multiple interfaces:

- Command-line management using PL/SQL procedures
- Oracle Data Safe
- Oracle Database Security Central

These options support enabling SQL Firewall, creating and managing allowlists, and managing violation logs. The best fit usually comes down to whether the environment favors scripted automation or visual, guided workflows.

Deciding between Database Firewall and SQL Firewall

Database Firewall operates as a network-based SQL firewall that monitors SQL traffic for both Oracle and non-Oracle databases. For non-Oracle databases, or Oracle Database versions earlier than 26ai, Security Central's Database

Firewall is the only option among these two approaches. For Oracle AI Database 26ai and later, use the following decision factors to determine which approach best fits the requirements:

Performance impact: With a network-based Database Firewall, the database server sees no added CPU load because SQL analysis runs on the firewall host. In host monitor and out-of-band deployment modes, Database Firewall introduces no database-server performance overhead. In proxy mode, minor network latency may appear depending on where the firewall sits relative to the application and database tiers. SQL Firewall introduces minimal observable latency but does add minimal CPU overhead, typically about 1.0% to 1.5%, depending on workload and configuration.

Deployment options: Database Firewall runs on dedicated hardware or a virtual machine and supports proxy, host monitor, and out-of-band modes. Only proxy mode can block traffic inline, and it requires client connection string changes to route traffic through the proxy. SQL Firewall is built into the database kernel, requires no client configuration changes, and is often the better choice when blocking is the primary use case or when client reconfiguration is impractical.

Scope of inspection: Database Firewall inspects SQL traversing the network. It is unaware of database synonyms, internal scheduled jobs, and stored procedure executions occurring within the database. For local connections, host monitor mode can observe activity but cannot block it. SQL Firewall runs inside the database with full access to session context and metadata, enabling comprehensive monitoring and blocking, including SQL generated internally. If the application server and database run on the same host, SQL Firewall is the appropriate choice.

Processing of encrypted traffic: Database Firewall needs additional configuration to handle encrypted SQL traffic, typically involving SSL proxy or certificate management. With SQL Firewall, this is seamless because it inspects traffic after the database has already decrypted it.

Transaction integrity: Blocking SQL at the network layer with Database Firewall can disrupt in-flight transactions because each SQL statement is handled as a standalone operation. SQL Firewall, by contrast, understands transaction boundaries and honors the ACID properties (atomicity, consistency, isolation, and durability), helping ensure blocked operations do not leave transactions in inconsistent states.

High availability: High availability for Database Firewall is achieved through redundant firewall appliances, with clients distributing connections or failing over based on connection parameters or third-party load balancers. SQL Firewall is integrated into Oracle Database and automatically leverages existing high availability architectures such as Real Application Clusters (RAC) and Data Guard, with no additional high availability configuration.

Separation of duties: Database Firewall can monitor database traffic without input or control from database administrators, supporting strict separation of duties requirements. SQL Firewall is part of Oracle Database and requires cooperation from database administrators to deploy and configure, which may not align with environments that require complete DBA exclusion from security policy enforcement.

Licensing costs: If the organization already licenses Security Central, there is no licensing tradeoff because SQL Firewall is included with Security Central. If the organization already licenses Oracle Database Vault, SQL Firewall is typically the better choice because it is included with Database Vault. Note that most Oracle Database cloud services include Database Vault by default. If the organization licenses neither Security Central nor Database Vault and is not using an Oracle Database cloud service, either Security Central or Database Vault must be procured to access these SQL injection protection capabilities.

Behavioral protection for AI-generated queries

Oracle AI Database 26ai SQL Firewall and Security Central address a growing challenge with AI workloads: the SQL generated by AI systems cannot be predicted or practically reviewed in advance. AI-generated queries, multistep agent tool chains, and rapidly developed AI-assisted applications can produce syntactically valid but risky SQL, including overly broad SELECT statements, unvetted JOIN operations, and exploratory schema queries that traditional access controls may not intercept.

Traditional database security assumes SQL is written by humans, reviewed during development, and relatively stable in production. AI-generated SQL breaks each of these assumptions. Queries are created dynamically, never reviewed by humans, and can shift unpredictably based on natural language inputs or agent reasoning chains.

Database firewalls learn legitimate query patterns during normal operation and then enforce those learned behaviors in real time. SQL that deviates from established patterns can be blocked before execution, including unauthorized table access, novel JOIN patterns, unusual aggregation operations, DDL attempts, or bulk data extraction queries. This approach can stop injection attacks that try to explore database schemas, such as `UNION SELECT * FROM all_tables`, prevent AI agents from pivoting into sensitive tables outside their normal scope, and block prompt injection attacks that generate destructive commands.

For AI workloads where SQL is inherently untrusted, whether generated by models, composed by automation, or written without human review, database firewalls turn query anomalies into clear enforcement boundaries. They help keep AI assisted applications and autonomous AI agents within learned, safe operational parameters regardless of code quality, prompt engineering robustness, or credential exposure risk.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- **Oracle Database Security Central**: Comprehensive workshop covering Security Central deployment, policy configuration, and violation analysis.
- **Oracle SQL Firewall:** Hands-on lab for SQL Firewall configuration and management.
- **Managing SQL Firewall with Oracle Data Safe**: Lab 7 of the "Get Started with Oracle Data Safe Fundamentals" workshop covers SQL Firewall management through the Data Safe console.

Summary

SQL injection remains one of the most common and damaging ways attackers reach data, and its staying power is a reminder that awareness alone doesn't create consistent prevention. It's an application-layer flaw that appears when untrusted input is treated as executable SQL, letting an attacker run statements that read, modify, insert, delete, or destroy data. In multi-tier environments, the injected SQL typically runs with the application service account's privileges. Because those accounts often carry broad schema and procedure rights, a low-privilege attacker can suddenly operate with high-impact authority.

It's also important to treat SQL injection as an entry point to broader compromise, not just a one-time leak. Injection can enable theft, corruption, destruction, and lateral movement into other systems. A familiar failure pattern is that an application enforces access control in the middle tier, but injection through an input field or URL parameter slips around authentication and authorization. The recommended posture is defense in depth: upstream controls such as a web application firewall, paired with downstream controls that evaluate the SQL that actually reaches the database. Signature-based defenses are noted as limited against novel techniques, which is why database-layer controls matter for containing risk when application vulnerabilities inevitably exist.

Database-centric firewalls are presented as a practical access control approach: learn the normal SQL an application uses, then enforce an allowlist, blocking or alerting when incoming SQL deviates. Two options are highlighted. The network-based Database Firewall in Security Central operates at the network layer, can protect multiple heterogeneous databases from a centralized deployment, supports blocking in proxy mode, and supports monitoring in host monitor and out-of-band modes. The database-resident SQL Firewall built into Oracle AI Database 26ai runs inside the database kernel. The chapter's guidance favors `allowlists` over denylists because legitimate application SQL is finite, while malicious patterns are effectively unlimited.

SQL Firewall inspects incoming SQL statements that are subject to SQL Firewall policy. It also validates trusted connection paths using richer session context and metadata, helping counter evasion techniques such as encoded SQL, synonyms, and dynamically generated object names. Violations are captured in a dedicated view, and administration can be handled through PL/SQL procedures, Oracle Data Safe or Security Central. Decision guidance compares the two firewall approaches across performance, deployment practicality, inspection scope, encrypted traffic handling, transaction integrity with ACID awareness, high availability fit with RAC and Data Guard, separation of duties, and licensing considerations. The discussion also extends to AI workloads, where SQL is often dynamically generated. In that setting, learned behavioral boundaries help block unusual access patterns such as novel joins, schema exploration, bulk extraction, or destructive commands.

Once SQL injection risk is reduced, the conversation can move beyond stopping bad statements and toward a more interesting question: what valid, approved access should actually be allowed to show? The next chapter on Data-driven Authorizations brings readers into that more nuanced frontier, where policy shifts from a blunt gate into a finely tuned instrument that decides which rows and columns each user can access, and which information should remain just out of reach. It is a natural next step for teams that want defenses to feel less like a barricade and more like a well-designed map, guiding every query toward safe ground while keeping sensitive data safely within its proper boundaries.

.

Chapter 9

Data-Driven Authorization

The need for data-driven authorization

Organizations often need to control access to individual rows or columns of data within database tables. The access control mechanisms traditionally used to protect entire database objects cannot selectively restrict access to specific data rows, columns, or even cells within a table.

The rise of AI agents in business applications introduces a new dimension to this challenge. Agentic AI systems persistently attempt to access any data necessary to fulfill their assigned objectives. Without precise, fine-grained authorization controls, these systems can inadvertently retrieve sensitive or unauthorized information in response to prompts. As AI capabilities grow, enforcing granular data access restrictions becomes essential to protecting sensitive and regulated information from unintended disclosure.

Implementing data-level access controls directly within applications can be costly. Any changes to the security model typically require code modifications. For packaged or third-party applications, such updates may be impossible. Enforcing policies only at the application level can result in inconsistent protections, as different applications often rely on different security models. Direct database connections, whether through a SQL client, business intelligence tool, or Model Context Protocol (MCP) integration, can bypass application-level controls entirely.

By enforcing fine-grained access controls in the database, organizations centralize management and enhance security. Access policies applied at the database layer are enforced uniformly, regardless of which tool or application accesses the data. With several fine-grained access-control options in the Oracle Database, understanding the trade-offs helps organizations choose the right solution.

Understanding data-driven authorization

Data-driven authorization controls access at the row and column level. Beyond leveraging privilege grants and session context, data-driven controls can consider the data values themselves when determining access rights. A straightforward example is a table of employee information that includes a column indicating each employee's manager. A data-driven policy might restrict access to only records for a manager's direct reports, preventing the manager from seeing information about other employees. The policy is data-driven because it relies on the value in the manager column to make access decisions.

Today's applications are highly complex, often requiring authorization policies that consider user roles, user and organizational attributes, session information, and other contextual factors. Ensuring that access control checks are consistently implemented throughout an application is both challenging and difficult to maintain over time.

Access for users should be limited to the specific rows and columns required to fulfill their job responsibilities. However, when an administrator grants a database object privilege such as SELECT or INSERT to a user, that user receives access to all data within the table. Because database tables typically store far more information than any one user requires, this broad access poses a risk. For instance, a customer should only view their own records in a support application, not those of other customers. A help desk technician should access only tickets assigned to them, not all open tickets. Sales managers need visibility into sales opportunities for their direct reports but not for other sales staff. These scenarios exemplify fine-grained authorizations where application users must be restricted to only the relevant rows and columns.

Figure 9-1 illustrates a table containing a mix of sensitive and non-sensitive human resources data. As an employee, Nancy Greenberg should be able to see phone numbers, names, and manager names for everyone, while other sensitive data remains hidden. Nancy should see her own national identification number, and, as a manager, she should also be able to view the salaries of her direct reports.

Name	Manager	National ID	Salary	Mobile
Adam Fripp	Steve Stiles			650-123-3234
Neena Kochar	Steve Stiles			650-124-8234
Nancy Greenberg	Neena Kochar	000-51-4569	120300	515-123-4567
Luis Popp	Nancy Greenberg		69000	515-123-4234
John Chen	Nancy Greenberg		82000	515-123-8181
Daniel Faviet	Nancy Greenberg		9000	515-123-7777

Figure 9-1: Fine-grained Access Control separates rows and columns within a table

Challenges with application-layer authorization

To implement access control policies on the EMPLOYEE table in the HR sample schema, as shown in Figure 9-1, developers typically write complex authorization code to regulate which rows and columns users can access under various conditions. This process usually requires a dedicated schema or set of tables to store user and role information, which the application uses to generate user-specific SQL queries. This approach introduces several key challenges.

First, developers must write substantial and intricate code and apply it across all contexts where users require access. Second, each application must independently replicate authorization rules because the database does not inherently enforce these policies. Third, direct-access tools such as analytical platforms, SQL*Plus, AI agents, and MCP connections bypass application logic, gaining unrestricted access to data.

Oracle Database overcomes these obstacles with technologies known collectively as data-driven authorization. Centralizing application authorization controls in the database simplifies and accelerates application development and provides a consistent set of policies to define and maintain.

Oracle Database data-driven authorization technologies

Oracle introduced Virtual Private Database (VPD), an automated predicate-based row-filtering security technology, more than 25 years ago in Oracle Database 8i. At that time, it was the only database with this capability. The next version of the database, Oracle Database 9i, introduced Oracle Label Security (OLS) to automatically filter rows based on user and data labels. Real Application Security (RAS), introduced in Oracle Database 12c, is also declarative like OLS, simplifying the development of secure applications. Each of these technologies is discussed in detail in the sections that follow.

Among the data-driven access control mechanisms, VPD and RAS are both features of Oracle Database Enterprise Edition. Label Security requires a separate license for on-premises databases but is included in all but the most basic levels of cloud database services. For details about licensing in a specific database package, consult the Oracle Database Features and Licensing application at https://apex.oracle.com/database-features/.

Virtual Private Database

Virtual Private Database enforces fine-grained row and column-level security by leveraging a PL/SQL function created by the database administrator or security team. This function is tied to a table, view, or synonym through a VPD policy. When a SQL statement targets the protected object, the function runs automatically, appending a WHERE clause to ensure only authorized rows and columns are returned.

For example, consider the following SQL query:

```
SELECT * FROM hr.employees;
```

The VPD function might rewrite the SQL statement to:

```
SELECT * FROM hr.employees WHERE department <> 'SUPER SECRET DEPARTMENT';
```

VPD allows for differentiated access, enabling users to view all employee records but update only their own, for instance. Sensitive columns can be controlled so that only authorized users see actual values, while others receive nulls.

Because VPD enforces policies at the database level, access rules are consistently applied across standard database access paths. For instance, as shown in Figure 9-2, the user can access HR.EMPLOYEES table data only for `DEPARTMENT_ID` equal to '80', and `SALARY` data is hidden by setting it to NULL. Running an unrestricted query still results in only permitted data being returned.

```
SQL> SELECT last_name, email, department_id, salary FROM hr.employees;

LAST_NAME    EMAIL                DEPARTMENT_ID    SALARY
------------ -------------------- ---------------- ------
Hunold       AHUNOLD@EXAMPLE.COM  80
Ernst        ERNST@EXAMPLE.COM    80
Austin       DAUSTIN@EXAMPLE.COM  80
Pataballa    VPATABAL@EXAMPLE.COM 80
Lorentz      DLORENTZ@EXAMPLE.COM 80
```

Figure 9-2: VPD provides automatic & transparent row-level security

Rapidly changing authorization requirements, combined with the fact that VPD policies are implemented in PL/SQL rather than declarative SQL, can significantly increase complexity over time. These challenges highlight the need for a more scalable and consistent approach to access control, which leads to the next topic: controlling access to data using data labels.

Oracle Label Security

Another method of fine-grained access control is based on data labeling, where each row is assigned a label that represents its sensitivity or classification. This model is commonly used in government and regulated corporate environments, where information may be classified as TOP SECRET, CONFIDENTIAL, or INTERNAL USE ONLY. Access to data is then governed by a user's clearance level (which is also represented as a label), ensuring that users can only view data at or below their authorized classification. When labels are attached at the row level, the database can inherently determine which data is sensitive and enforce access restrictions accordingly. This approach is also referred to in some data privacy regulations as enforcing `Restriction of Processing`, where access and the ability to process data is limited based on data classification and user authorization. OLS supports the technical enforcement of this legal requirement.

Oracle Label Security simplifies assigning labels to data and users and enforces access control based on those labels. In an OLS-protected table, every row is associated with a label that reflects the sensitivity of the data. As an added security measure, the column containing the label values is hidden. Labels can be explicitly assigned based on business logic or automatically derived using default labeling rules tied to the application context or user session that inserts the data. The effective label of a user session is calculated using multiple factors, including the label assigned to the user, session attributes, and the type of database connection. In this sense, labels act as an extension to traditional database privileges and roles. A label can be associated with a database user, and starting with Oracle

Database 12c Release 2, a label can also be associated with application users supported by Real Application Security (discussed later in this chapter).

Label components and structure

The OLS label model is flexible and can represent virtually any data classification scheme. Each label is composed of up to three components:

- **Level**: A mandatory hierarchical classification, with exactly one level per label
- **Compartment**: Optional categories used to further restrict access, with zero or more compartments per label
- **Group**: Optional hierarchical structures used to model organizational or geographic boundaries, with zero or more groups per label

Every label includes a level, ordered from lowest to highest, representing the overall sensitivity of the data. Optional components called compartments and groups may segregate information based on attributes such as projects or departments. Compartments are typically used to isolate information related to specific domains, such as finance or pricing. Groups enable multi-dimensional, hierarchical authorization models, such as regional or organizational structures.

Figure 9-3: Example OLS policy showing Label Security components: Levels, Compartments, Groups shows an example of a global retail organization classification scheme using all three label components. Three levels are defined: `HIGHLY SENSITIVE, SENSITIVE`, and `PUBLIC`. Only users with a `HIGHLY SENSITIVE` label can access unreleased quarterly financial data. A FINANCE compartment grants access to those needing to see financial data. Upcoming pricing changes are also sensitive, so a `PRICING` compartment protects pricing data. Groups represent the retail stores' geographic hierarchical structure. This combination of levels, compartments, and groups enables precise, scalable, and policy-driven access control across complex data environments.

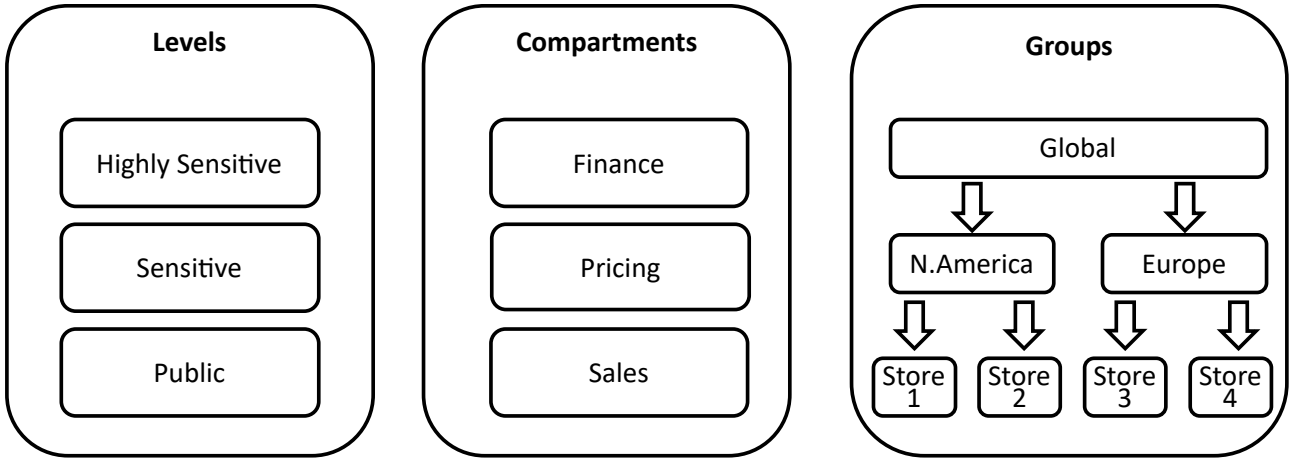


Figure 9-3: Example OLS policy showing Label Security components: Levels, Compartments, Groups

Labels are represented by the three components separated by a colon (:) as shown in Figure 9-4.

```
LEVEL:COMPARTMENTS:GROUPS
```

Figure 9-4: Label component organization

A data label representing sensitive finance data for Store 1 would be represented as `SENSITIVE:FINANCE:STORE1`. Similarly, pricing data for Store 1 would be labeled `SENSITIVE:PRICING:STORE1`. In this model, a Store 1 marketing

manager who requires pricing information for creating weekly sales newspaper inserts and promotional emails would be assigned a user label that includes the PRICING compartment.

Evaluating labels: Groups, Compartments, and Levels

To illustrate how labels are evaluated, the following examples begin with the group component and then introduce compartments and levels.

In the label design shown in Figure 9-5: Using the Group component in Label Security, users in each store can only see data regarding their store due to the restrictions imposed by the group label component. Users assigned the STORE1 group label can only view data labeled for STORE1. A North America regional director, however, may be assigned the N. AMERICA group label. Because groups are defined hierarchically, this allows access to all subordinate groups, such as STORE1 and STORE2. This hierarchical group structure enables centralized data storage while ensuring users only see data relevant to their organizational scope.

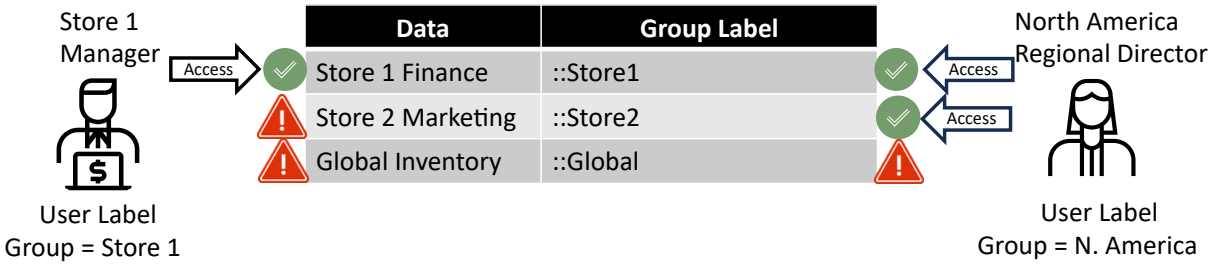


Figure 9-5: Using the Group component in Label Security

The retail company required that financial and pricing data be visible only to staff with a legitimate business need. To enforce this, FINANCE and PRICING compartments were created, and the corresponding data records were labeled accordingly. User access was then reviewed and aligned with these compartments. The Store 1 marketing manager was granted access to pricing data to produce weekly sales materials, while the Store 1 manager was granted access to both pricing and financial data for Store 1. However, neither role was permitted to access financial data belonging to the N. AMERICA group.

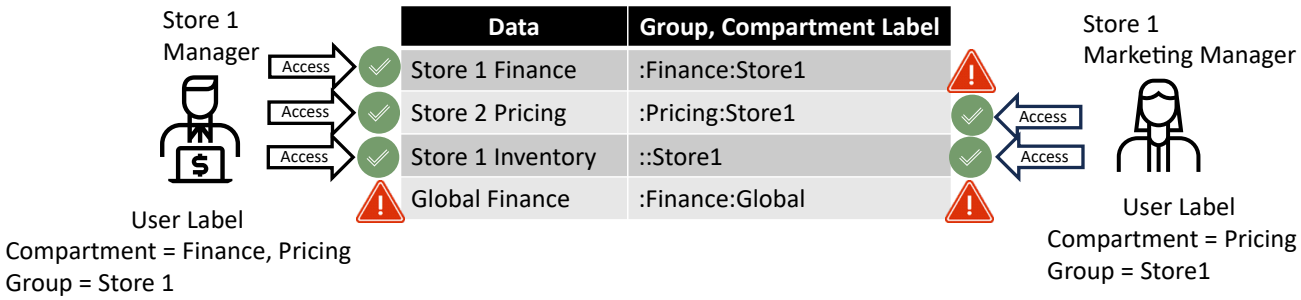


Figure 9-6: Using the Compartment and Group components in Label Security

As a public company, the global financial data and reports are classified as highly sensitive. All data is considered sensitive, but global financial data is specifically highly sensitive within the FINANCE compartment and GLOBAL group. Only users whose labels reflect the level HIGHLY SENSITIVE with the FINANCE compartment and the GLOBAL group may see this information.

User access conditions

User access to data requires that their user label meets the following conditions in all three components:

- **Level**: The user's label must be at the same or a higher level than the data's label. For instance, a user labeled `SENSITIVE` can access `SENSITIVE` or `PUBLIC` data, but not `HIGHLY SENSITIVE` data.
- **Compartment**: The user's label must contain every compartment listed on the data. Data labeled with the `FINANCE` compartment requires the user's label to include FINANCE. If the data label specifies both FINANCE and PRICING, the user's label must include both compartments.
- **Group**: The user's group label must match at least one group from the data label or encompass it hierarchically. For example, if the data is labeled `STORE1`, users with `STORE1`, `N. AMERICA`, or `GLOBAL` group labels can view the data.

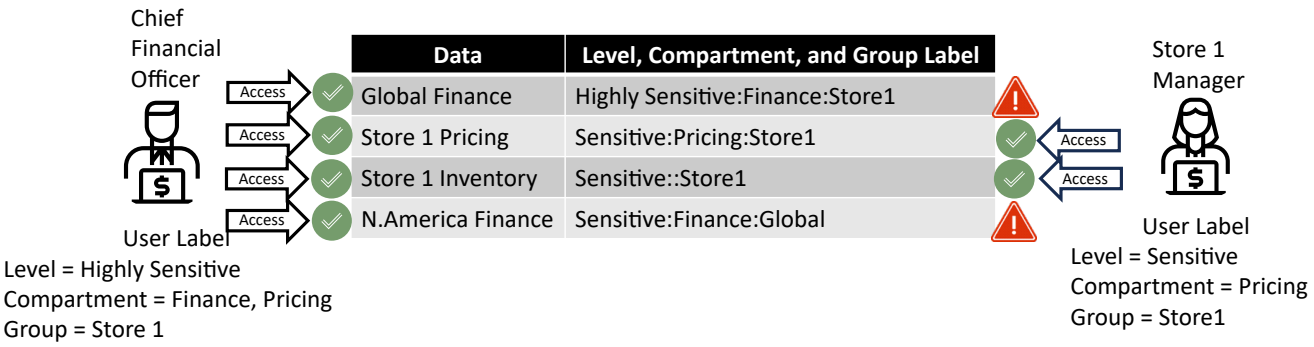


Figure 9-7: Using the Level, Compartment, and Group components in Label Security

Label Security use cases

Label Security was originally developed to meet the stringent data labeling and mandatory access control requirements of government, military, and intelligence agencies. However, commercial organizations have also found that Label Security simplifies data access authorization. For example, in the commercial scenario described above, each retail store previously operated its own applications and databases, making consolidated reporting challenging and maintaining parallel systems costly. By consolidating data into a single database, Label Security enables robust access controls while reducing operational complexity.

Another common commercial application relates to data protection laws, such as the European Union's General Data Protection Regulation (GDPR), which require restricting data processing based on user consent. In this case, groups and compartments can record the types of processing a data subject has consented to, which is reflected in the data label. Applications then set a user session label, ensuring that records the application should not process remain inaccessible.

A major advantage of Label Security over VPD or RAS is a complete authorization infrastructure that provides automatic mediation, enforcement, and simplified administration. Label Security uses hierarchical user and data labels comprised of levels, compartments, and groups that parallel real-world scenarios without requiring administrators to write PL/SQL policy functions. Once the model is configured for user and data labels, access control is consistently enforced across all interfaces, eliminating the need for explicit configuration on a per-table basis. Label Security is recommended when access control logic can be defined using user and data labels.

Real Application Security

Previously, we saw that VPD manages data access at the row and column level. However, in most modern applications, end users do not interact directly with the database and typically do not have individual database accounts. Instead, a single database account represents the application, and it queries and updates data on behalf of all end users, who are managed by the application itself.

Because the database is not aware of individual end-user identities, the application must handle per-user access control. This approach requires additional development effort and often results in inconsistent policy enforcement, particularly if other tools, clients, or applications can access the database directly and bypass the application layer.

To address these challenges, Oracle introduced Real Application Security in Oracle Database 12c as a next-generation access control framework within the database. RAS allows two-tier and three-tier applications to define, provision, and enforce access requirements declaratively in the database layer, providing a simpler, more maintainable framework that can replace VPD in nearly all scenarios.

RAS is built on a policy-based authorization model that recognizes application users, privileges, and roles in the database, controlling access to both static and dynamic groups of records that represent business objects. This approach resolves the scalability, maintainability, and security limitations associated with complex VPD policies and supports common application patterns such as master-detail tables and temporary assignments.

Application user identity propagation

RAS securely propagates the identity of the application end user to the database. Like VPD or OLS, RAS enforces access control policies at the database level, regardless of which application accesses the data. The application can initiate multiple application user sessions and switch between them using a single database connection from a connection pool. Note that the application user does not have its own schema to store data objects or a dedicated connection to the database as a regular database user would. RAS can also enforce access control policies on traditional database users.

Data Realms and access control lists

RAS supports fine-grained column and row access restrictions, similar to VPD, but uses a more declarative syntax. Administrators define data realms for table rows, each using a WHERE clause-like syntax to identify subsets of data. Access control lists (ACLs) are then attached to each data realm, specifying which users or roles can perform which operations on data within that realm.

As illustrated in Figure 9-8: Example illustrating the data access controls with Real Application Security, three example data realms might be created for the same table: a public data realm covering all employee records, a self-data realm (a dynamic realm that protects each user's own record), and a manager data realm (another dynamic realm that covers employees reporting to the user).

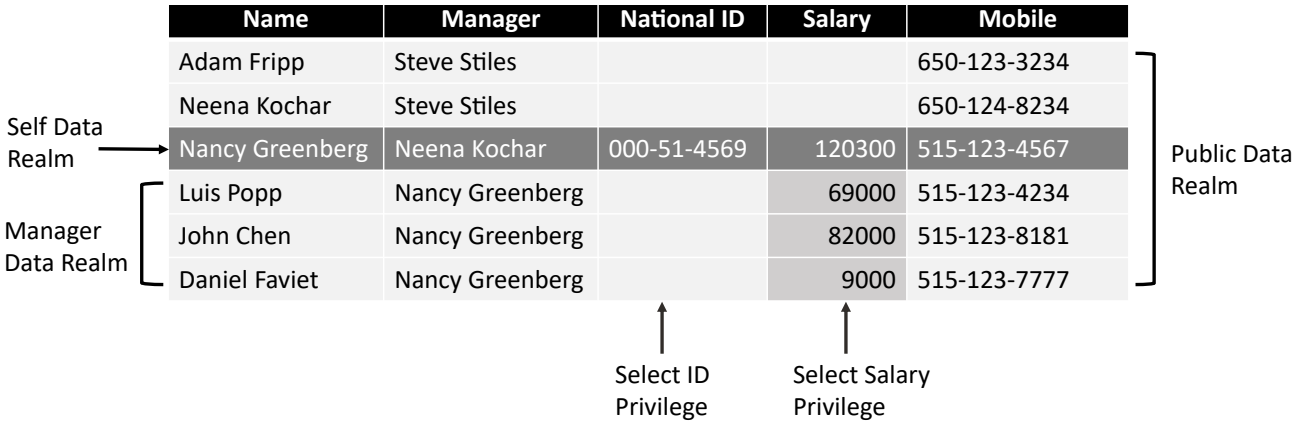


Figure 9-8: Example illustrating the data access controls with Real Application Security

For each data realm, an access control list defines who can access that data realm and under what conditions. For the public data realm, all employees have only the `SELECT` privilege.

All non-protected data should be visible to any employee. Because `SSN` and `Salary` data are protected and not included in the ACL for the public data realm, data from those columns does not appear when queried by someone with only public access. The self-data realm is associated with an ACL to view `SSN` (Authorize SSN) and Salary (Authorize Salary) data so that an employee can view their own sensitive data.

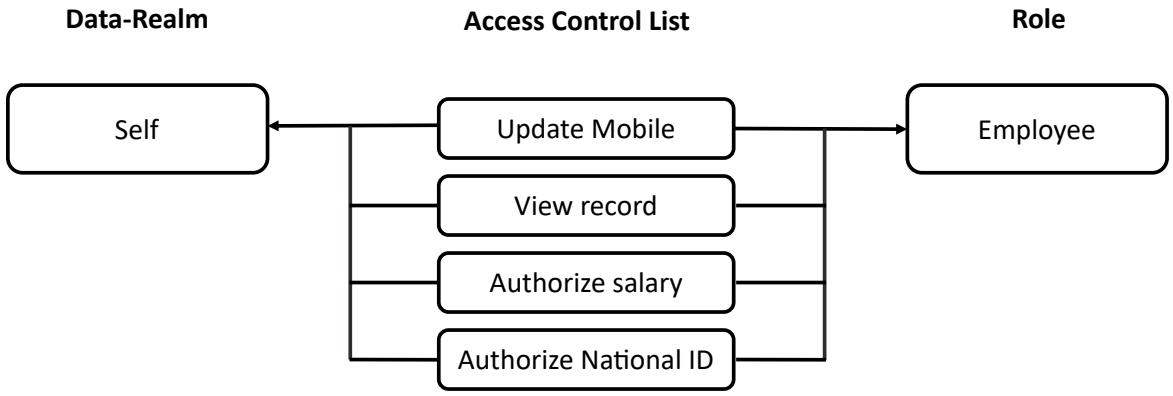


Figure 9-9: Example of a RAS Access Control List (ACL)

Managers can view their direct reports' salary information via the manager data realm, which is governed by an ACL permitting access to salary data. Both self and manager data realms are dynamic, returning the relevant rows and columns according to the application user's session identity.

Application-defined privileges

RAS also allows the database to enforce additional security policies unique to each application. Applications can define privileges beyond standard SELECT, INSERT, UPDATE, and DELETE operations, such as those for approving vacation requests or processing invoices. Administrators may assign access to sensitive columns such as salary or SSN only for users holding specific application-defined privileges (for example, UPDATE_SALARY or VIEW_SSN).

Minimal privilege design

Connecting through the RAS Java interface offers more security than traditional methods, which often require application accounts to have all privileges necessary for every user operation, including those needed for

maintenance and installation. These highly privileged accounts become prime targets for attackers. RAS enables the use of minimally privileged application accounts, so even if a compromise occurs, access to sensitive data remains protected. RAS realms, ACLs, and policies can be fully managed using the RAS PL/SQL API.

With built-in support for propagating application user sessions to the database, RAS allows security policies on data to be expressed directly in terms of the application-defined users' roles and data operations. The RAS security model enables uniform specification and enforcement of access control policies on business objects irrespective of the access path. Using declarative access control policies on application data and operations, RAS enforces security close to the data and enables end-to-end security for both three-tier and two-tier applications. The declarative model for RAS is much simpler to maintain and extend than VPD.

Choosing the right technology

Virtual Private Database is the most straightforward access control feature to begin using. However, when VPD PL/SQL policies become complex, the code to create the WHERE predicate clause becomes difficult to manage. If the access control policy is simple and will remain that way, VPD is an appropriate choice.

Real Application Security is a more powerful data-driven access control technology. It is enterprise-ready with many enterprise patterns built in. It can serve as a direct replacement for VPD, working directly with database users, roles, and objects. If an organization is developing a new application or has the ability to modify existing application code, RAS provides a robust data security framework that separates access control policies from the applications. The application simply invokes the RAS APIs to connect application users to the database session and set any needed context variables.

Label Security is an excellent choice because the logic to evaluate label precedence is handled automatically. However, work is required to manage and maintain user and data labels. Both VPD and RAS provide row and column-level control, while Label Security operates only at the row level.

Organizations should evaluate these technologies based on several factors: the complexity and expected evolution of access control requirements, the ability to modify application code, the need for application user identity propagation, and whether the access control model can be represented using hierarchical labels. For new applications with complex authorization needs, RAS is typically the best choice. For scenarios where data classification naturally maps to hierarchical labels, Label Security offers the simplest administration. For simple, stable access control requirements on existing applications that cannot be modified, VPD remains a viable option.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Protecting row and column data with Virtual Private Database
- DB Security - Label Security

Summary

Organizations increasingly need access control that reaches beyond whole tables down to individual rows and columns, especially as AI agents in business applications repeatedly try to fetch whatever data might help complete a task. When authorization lives only in application code, changes get expensive, enforcement varies from app to app, and controls can be sidestepped through direct database paths like SQL clients, analytics tools, AI agents, or Model Context Protocol (MCP) connections. Centralizing fine-grained authorization in the database keeps policies consistent, no matter how the data is accessed.

Data-driven authorization is presented as a more grounded way to make access decisions: evaluate privileges and session context, but also evaluate the data values themselves. In an employee table, for example, a manager should only see direct reports because the policy checks the manager column to decide which rows qualify. The same idea maps cleanly to other common expectations: customers should see only their own service records, help desk technicians only the tickets assigned to them, and sales managers only opportunities tied to their teams. A detailed HR example makes the point vivid in a mixed-sensitivity table: Nancy Greenberg can see general contact and manager information for everyone, can see her own national identification number, and, as a manager, can see salaries for her direct reports, while other sensitive values remain out of view.

Three Oracle Database technologies form the core toolbox for enforcing this kind of fine-grained control: Virtual Private Database (VPD), Oracle Label Security (OLS), and Real Application Security (RAS).

- **VPD** attaches a PL/SQL function, written by an administrator or security team, as a policy. The database rewrites queries by adding a predicate that filters rows and can return `NULL` for unauthorized columns, enforcing controls even when the query itself is unrestricted. It's described as the most straightforward on ramp, with the tradeoff that policy logic can become harder to manage over time because it lives in PL/SQL rather than declarative SQL.
- **OLS** labels each row with a classification and evaluates access using levels, compartments, and groups. The retail example uses levels like `HIGHLY SENSITIVE, SENSITIVE`, and `PUBLIC`; compartments such as `FINANCE` and `PRICING`; and hierarchical groups like `STORE1`, `N. AMERICA`, and `GLOBAL`, with explicit access conditions for each dimension.
- **RAS**, introduced in Oracle Database 12c, propagates application end-user identity into the database, defines data realms and access control lists declaratively, supports application-defined privileges, and enables minimal-privilege application accounts to reduce the blast radius of compromise.

Selection guidance comes down to fit and friction: consider policy complexity and expected change, constraints on modifying application code, whether end-user identity propagation is required, and whether the authorization problem naturally maps to a label model. In general, RAS is positioned as a strong match for new applications with complex authorization, OLS fits clean classification models, and VPD works well for simpler, stable policies, especially when applications can't be modified.

With data driven authorization in place, the narrative naturally widens from who can see which rows and columns to how sensitive values should look when they do appear on screen. The next chapter on Data Masking is where policy meets presentation, turning raw secrets into carefully curated shadows so teams can keep moving fast without letting regulated details spill into the wrong hands. After that, Encryption takes the stage as the final vault door, protecting data at rest and in motion so that even if something is copied, intercepted, or mishandled, what remains is unreadable noise to anyone without the keys. Keep reading, because these two chapters complete the arc from access control to safe exposure, to durable protection, and together they form a practical playbook for building systems that stay trustworthy under real world pressure.

Chapter 10

Masking Sensitive Data

Why masking sensitive data matters

Applications handle large amounts of sensitive information, including personally identifiable information (PII), financial records, healthcare data, and proprietary business details. Organizations face a fundamental challenge: extracting maximum value from data while protecting it from unauthorized access or misuse. Limiting the exposure of sensitive data to users who do not need to see it is a challenge that grows more acute as data proliferates across development, testing, analytics, and operational systems.

The business case for using actual production data is compelling. Development teams need realistic datasets to build and test applications that function correctly under real-world conditions. Marketing and customer service teams deliver better outcomes when they can analyze customer behavior patterns and interaction history. Call center representatives resolve issues faster when they can review purchase histories and account details. Fraud analysts require actual transaction patterns to train detection models. Each of these scenarios delivers clear business value, but each also increases the risk of exposing sensitive information to unauthorized parties or unintended uses.

Most organizations create multiple copies of production databases for use in testing, development, user acceptance testing, and training. These non-production copies may also be shared with partners or third-party vendors for specialized analytics or fraud detection. Each additional copy raises the probability that sensitive information could be compromised through misconfigured access controls, insider threats, lost devices, or external attacks. As organizations embrace artificial intelligence and machine learning, these same environments are increasingly used to train models and power retrieval-augmented generation (RAG) pipelines. Without proper controls, real PII can proliferate into systems that were never designed to protect it.

Data masking addresses this challenge directly. By transforming sensitive data into realistic but fictitious values, organizations protect customer privacy while preserving the utility of data for legitimate business needs. Masked data holds little or no value to attackers, yet it remains valuable for development, testing, analytics, training AI models, and operational support. When testing a point-of-sale application, development teams can use actual inventory levels and store configurations while substituting realistic but fictitious values for customer names, email addresses, and payment details. This approach maintains referential integrity and application functionality while substantially reducing the risk of data breaches.

Data masking, combined with data redaction and data subsetting, forms an integral part of a comprehensive data privacy strategy. These techniques help organizations enforce the principle of least privilege, satisfy anonymization and pseudonymization requirements, and meet compliance obligations under regulations such as PCI-DSS, EU GDPR, CCPA, and India's DPDPA. In an environment where data privacy and security are both regulatory requirements and competitive differentiators, these strategies enable organizations to manage sensitive data confidently while supporting business objectives.

Understanding data masking

Data masking protects sensitive information by replacing real data with realistic but fictitious values. The fictitious data preserves the appearance and structure of the original, making it suitable for development, testing, and analytics, but it has little or no value to an attacker. Data masking is sometimes referred to as redaction, scrambling, or obfuscation—and is sometimes (informally) labeled anonymization—though important technical distinctions exist among these terms.

Unlike encryption, data masking is not designed to be reversed. Once data has been masked, the original values are typically not recoverable through standard administrative processes. This irreversibility is a strength when the goal is to permanently remove sensitive information from non-production environments. Masked data should remain consistent across related tables and respect referential integrity rules. For example, if a customer ID appears in both a customer master table and an orders table, the masked value must be identical in both locations to preserve the relationships that applications depend on.

Data masking strategies fall into two broad categories: static and dynamic.

Static data masking modifies the stored data itself, permanently replacing sensitive values with masked alternatives. This approach is used when sensitive data must be removed from non-production environments such as development, testing, training, or analytics systems. Static masking may also be applied when sharing data with external organizations for purposes like fraud analysis or clinical research. After static masking is complete, all users who access the data see only the masked values because the original unmasked data no longer exists in that environment.

Dynamic data masking—referred to in Oracle Database Security documentation as Data Redaction—prevents the proliferation the proliferation of sensitive data by masking values as they are retrieved, while leaving the stored data unchanged. With dynamic data masking, the results returned to a user depend on the masking policy in effect. Some users may see masked values, while others with appropriate authorization see the original data. Dynamic masking is typically applied in production environments where different users require different levels of access to sensitive information.

Oracle Database provides comprehensive support for both static and dynamic masking. Organizations can use Oracle Data Safe or Oracle Data Masking and Subsetting for static masking. For dynamic data masking, Oracle offers Data Redaction (included with Oracle Advanced Security), Virtual Private Database, and Real Application Security.

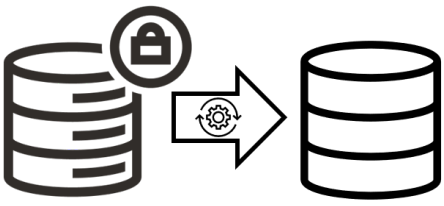
Static data masking

Static data masking creates datasets in which the original sensitive data no longer exists. After masking is applied, it does not matter who attempts to access the data. Everyone sees only the masked version, and the original values are gone. This permanence makes static masking the appropriate choice for non-production environments where developers, testers, and analysts need realistic data but have no legitimate need to access actual customer, employee, or financial details.

Databases masked in this way can be provisioned to development, testing, and business analytics teams while minimizing the exposure of sensitive information. Organizations can also use statically masked data when sharing datasets with external partners, vendors, or research institutions.

Production data

Last Name	SSN	Salary
AGULAIR	203-33-3234	60,000
BENSON	323-22-2943	40,000



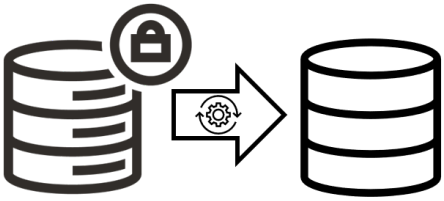
Masked non-production data

Last Name	SSN	Salary
SMITH	148-92-3857	40,000
DOE	562-99-8392	60,000

Figure 10-1: Example static data masking illustrates static data masking. In this example, actual employee names from a production database are replaced with realistic but fictitious names. Real Social Security numbers are transformed into random values that follow the correct format. Salary information is shuffled so that the overall payroll total remains accurate for aggregate analysis, but no individual record reveals actual compensation data. The production database retains sensitive information and requires full security controls. The non-production database, containing masked data, carries less inherent risk and can be managed with reduced security overhead.

Production data

Last Name	SSN	Salary
AGULAIR	203-33-3234	60,000
BENSON	323-22-2943	40,000



Masked non-production data

Last Name	SSN	Salary
SMITH	148-92-3857	40,000
DOE	562-99-8392	60,000

Figure 10-1: Example static data masking policy

Organizations can perform static data masking using services external to Oracle Database, such as Oracle Data Safe or Oracle Data Masking and Subsetting.

Use cases for static data masking

Static data masking supports several important scenarios:

Reduce risk and simplify compliance: By masking sensitive personal data in non-production environments, organizations lower the probability and potential impact of a data breach. Less sensitive data in the environment means less to steal and less damage if a breach occurs. In some cases, masking can remove entire systems from the scope of privacy regulations such as GDPR and CCPA, simplifying compliance audits and reducing the administrative burden of managing regulated data.

Protect non-production environments: Developers and testers need realistic data to validate that applications perform correctly under production-like conditions. Data masking enables teams to work with real-world data structures, volumes, and relationships without exposing sensitive information. This approach also helps organizations meet regulations that explicitly prohibit the use of actual sensitive data in non-production environments, such as certain healthcare and financial industry standards.

Save time and reduce costs: Cloning a production database and applying masking produces a realistic test or development environment far more quickly than building synthetic datasets from scratch. Teams avoid the expense and complexity of creating and maintaining artificial data that may not accurately reflect production patterns. Masked production clones provide high-fidelity test data with minimal ongoing effort.

Secure data sharing and third-party analytics: When organizations need to share data with vendors, partners, or research institutions, static masking protects privacy and security. In specialized scenarios such as healthcare research or financial fraud analytics, reversible masking techniques can provide useful data for analysis without exposing sensitive details. Analysts can derive insights from masked datasets without ever accessing the original information.

Support training and education: Data masking allows students, trainees, and new employees to work with realistic data volumes and structures in educational settings or training programs without exposing private information. This use case is particularly important in industries like healthcare and finance, where professionals must learn to work with production systems before handling actual customer data.

Data Redaction

Dynamic data masking, referred to in Oracle Database Security as Data Redaction, prevents the proliferation of sensitive data by masking values as they are retrieved from the database, while leaving the stored data unchanged. Unlike static masking, which permanently transforms the data on disk, dynamic masking applies transformations at query time based on policy conditions. The result delivered to a user depends on the masking policy in effect: some users may see masked values, while others with appropriate permissions see the original data.

Dynamic masking is typically used in production environments where different users require different levels of access to sensitive information. The database enforces masking policies transparently, without requiring changes to application code or queries. This approach provides consistent protection across all applications and tools that access the database, eliminating gaps that can arise when security logic is scattered across application layers.

Figure 10-2: Example Data Redaction illustrates Data Redaction in action. A data analyst and a human resources partner both query the same employee table. The data analyst needs to see employee last names for reporting purposes but does not require access to full Social Security numbers or actual salary details. The database applies a redaction policy that hides part of the Social Security number and replaces the salary with a fixed value when responding to the analyst's query. The HR partner, who is authorized by policy to view full employee data, sees the complete Social Security number and actual salary. The data stored in the table remains unchanged. Only the query results are transformed based on policy.

Figure 10-2: Example Data Redaction policy

Oracle Database enables dynamic masking through several mechanisms: Data Redaction (part of Oracle Advanced Security), Virtual Private Database, and Real Application Security. Each approach offers different capabilities and trade-offs.

Use cases for Data Redaction

Data Redaction addresses several important scenarios:

Control access to specific data elements: Data redaction enables organizations to hide sensitive columns as data is delivered to database clients, ensuring that only users with a legitimate business need can view certain information. For example, business intelligence tools and SQL-based reporting systems rarely need to display credit card numbers, birth dates, or taxpayer identifiers in full. Redaction policies can mask these columns while allowing aggregate analysis and reporting to proceed normally. This approach is also valuable in artificial intelligence applications that use retrieval-augmented generation with relational data, where sensitive details should not be fed into language models or training pipelines.

Minimize access based on policy conditions: Organizations can configure dynamic masking to redact data when access conditions fail to meet policy requirements. For instance, a policy might prevent users from viewing sensitive columns when connecting from outside approved network segments, accessing the database outside normal business hours, or querying from applications that have not been explicitly authorized. This conditional masking provides an additional layer of defense in depth.

Protect sensitive data in legacy applications: Many legacy applications were designed before modern privacy standards took effect and may display sensitive information even when users do not need to see it. Updating application code to implement fine-grained access controls is often difficult, expensive, or impossible when source code is unavailable or the application is provided by a third-party vendor. Data redaction allows organizations to hide

or transform sensitive data at the database layer, reducing risk without modifying application logic. This ensures consistent protection regardless of which application or tool accesses the data.

Support training and education: Data redaction allows trainees and students to work with production or production-like systems in educational or training programs without exposing private information. Redaction policies can be configured to mask sensitive columns for training accounts while allowing experienced staff to see full details.

Data Subsetting

Data Subsetting helps organizations meet data minimization requirements and reduce risk in non-production environments. When the full production dataset is not necessary for testing, development, or analytics, subsetting extracts a smaller, representative portion of the data. This approach reduces the amount of sensitive information at risk, lowers storage costs for database copies, and accelerates operations like backups, restores, and environment refreshes.

Subsetting allows organizations to extract only the data needed for a specific purpose. An analytics team might require only 20 percent of the production dataset, or only records collected within the last year, or only data specific to a particular geographic region. This targeted extraction provides teams with relevant, realistic data while keeping security risks and operational costs in check.

Data Subsetting is often combined with Data Masking in a single workflow. Organizations first extract the required subset from production, then apply masking to any sensitive information within that subset before provisioning it to non-production environments. This combined approach maximizes the business value of data, protects sensitive information, and avoids wasting resources on unnecessary storage and processing.

Together, Data Masking, Data Redaction, and Data Subsetting limit the opportunities for attackers to steal sensitive data, reduce the impact of a database compromise, and help organizations maintain compliance with privacy regulations such as PCI-DSS and EU GDPR.

Masking formats

Masking formats define how original data is transformed into anonymized values during the masking process. Oracle Database provides a comprehensive library of masking formats that range from simple transformations to sophisticated predefined patterns designed for common types of sensitive data. Simple masking formats are basic transformations that can be applied to data. Table 10-1: Simple masking formats describes several commonly used simple formats:

Masking format	Description
Fixed value	Replaces original values with a fixed value. For example, all email addresses might be replaced with "fake@no.co".
Random values	Replaces original values with random values, usually within a specified range or constrained by data type.
Random list	Replaces values in a column using a predefined list or array of values. For example, first names might be replaced with names randomly selected from a list of common names.
Substitution	Replaces values in a column with values selected from another table. This format is useful when masked values must maintain certain properties, such as valid postal codes for actual cities.
Encryption	Applies a cryptographic algorithm to mask the data. This type of masking is used when there is a need to reverse the masking later, such as when sharing data with a third party for analysis and subsequently recovering the original data after processing is complete.

Format preserving randomization	Randomizes data while preserving its structure and format. Replaces letters with letters and digits with digits, but keeps the data length, special characters, and the case (upper or lower) of characters unchanged. This technique is often used for data elements like national identifiers, postal codes, and license plate numbers.
Shuffle	Randomly reorders the values within a column so that statistical properties such as distributions and totals are maintained. Often applied to columns like age, gender, or salary.
SQL expression	Uses a custom SQL expression to generate masked values. Example: email addresses can be generated from columns containing first and last names. SQL expression masking is applied after other columns have been masked allowing masked email addresses to be constructed in the form "first_name.last_name@company.com" while maintaining consistency between the name and email columns. Also, can be used to mask data contained in large objects (LOBs), including BLOBs, CLOBs, and NCLOBs.

Table 10-1: Simple masking formats

Organizations commonly combine multiple simple transformations to create realistic masked data that follows the formatting and syntax rules appropriate for specific data elements. For example, a masked street address might be constructed by concatenating a four-digit random number between 1 and 9999, a random selection from a list of common road names such as Main, Elm, or Broadway, and a random selection from a list of street types such as Street, Road, Avenue, Boulevard, or Circle. The result is a fictitious but plausible street address.

In addition to simple formats, Oracle Database provides predefined masking formats for common types of sensitive data, including credit card numbers, telephone numbers, Social Security numbers, and other national identifiers. These predefined formats combine several simple transformations and often include additional processing to ensure that masked values appear realistic and follow the correct syntax rules. Figure 10-3: Predefined masking formats in Oracle Data Safe shows a selection of more than 50 predefined formats available in Oracle Data Safe. Predefined formats simplify the design and implementation of masking policies.

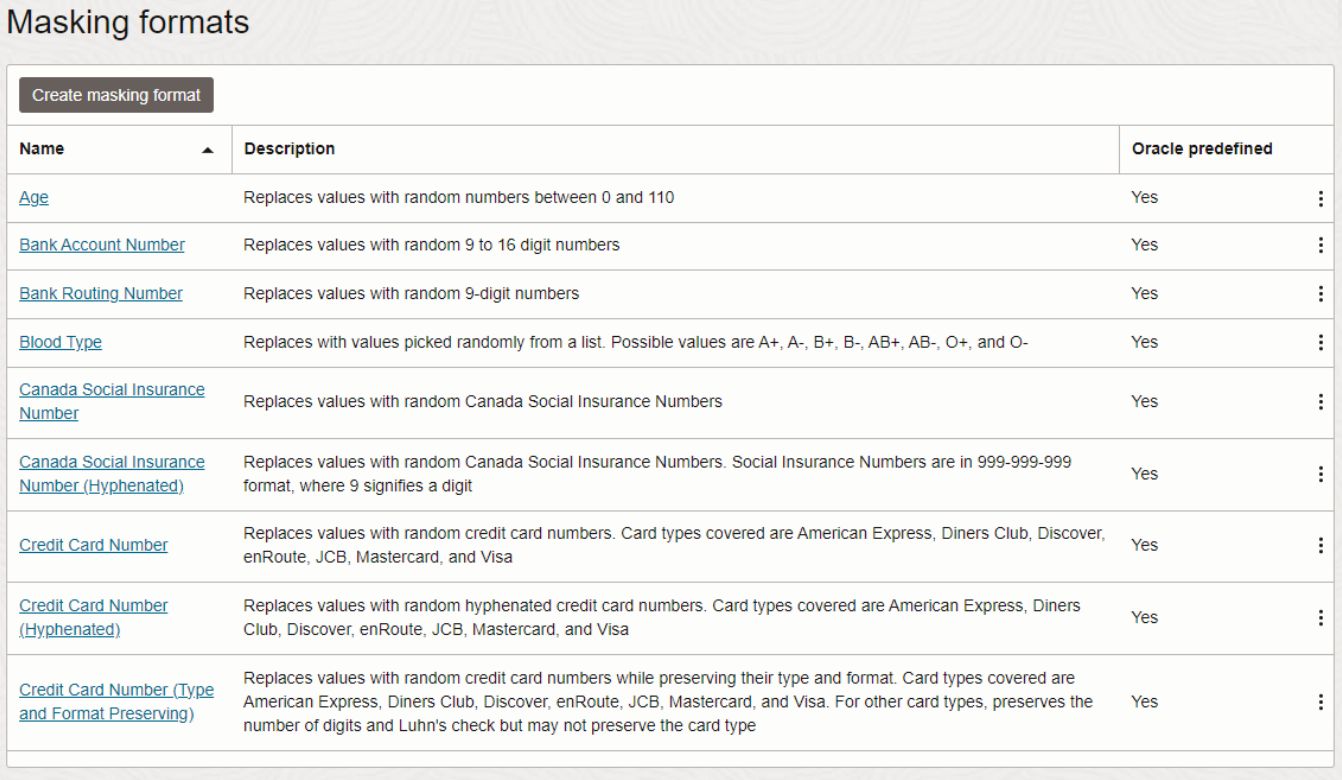


Figure 10-3: Predefined masking formats in Oracle Data Safe

Organizations are not limited to the formats Oracle provides. Custom masking formats can be created by combining existing formats or by defining entirely new transformation logic. If an organization uses unique data elements, such as proprietary account numbers or custom identifiers, it can create masking formats tailored to those requirements. This flexibility ensures that masked datasets are both secure and useful for their intended purposes.

Masking techniques

Masking techniques describe how transformations are applied to datasets in ways that preserve application functionality and data relationships. These techniques make it possible for complex applications to work correctly with masked data in non-production environments. Table 10-2 describes several important masking techniques.

Masking technique	Description
Conditional masking	Applies different masking formats to different rows based on specific conditions. For example, data about citizens from multiple countries can have unique masked national identifiers based on their country of origin. Similarly, credit card numbers can be masked while preserving their original card type and format.
Compound masking	Masks related data contained in multiple columns as a group. For example, city, state, and postal code can be masked together to ensure the values remain consistent. This prevents nonsensical results such as New York City located in Texas.
Deterministic masking	Produces the same masked output value for a given input value consistently across multiple databases, applications, or masking runs. Deterministic masking allows organizations to maintain referential integrity in a multi-database test environment. When the same customer ID appears in multiple databases, deterministic masking ensures it is masked to the same value everywhere.
Reversible masking	Encrypts sensitive data using a cryptographic key so that the original values can be recovered when necessary. This technique is useful when organizations must share masked data with third parties for business processing and later recover the original data after processing is complete. For example, healthcare data might be shared with researchers in masked form and then unmasked for follow-up patient care.
Shuffle masking	Randomly reorders values within a column, breaking the one-to-one mapping between related data elements. For example, a column containing salaries can be shuffled to break the association between employees and their compensation while preserving the overall distribution of salaries for statistical analysis.

Table 10-2: Summary table of available Masking techniques

Figure 10-4 demonstrates these techniques in action. Conditional masking produces different masked formats for national identifiers based on country codes. Random strings replace license plate numbers while preserving the original format. Deterministic masking ensures that employee names are masked consistently across all databases and masking operations. Shuffle masking disassociates health records from patient identities while preserving complex data types like diagnostic images.

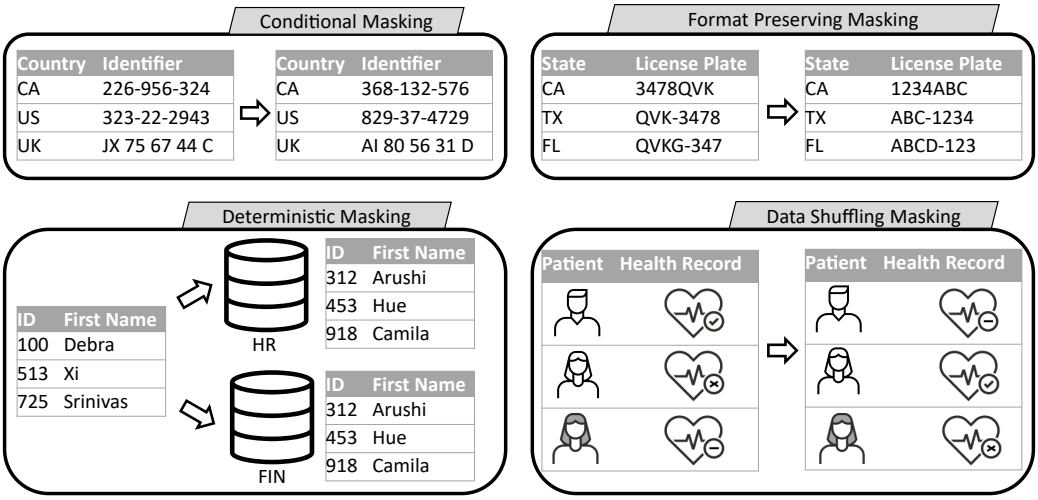


Figure 10-4: Masking formats and techniques in action

Implementing static data masking

Oracle Database provides two primary tools for static data masking: Oracle Data Safe and Oracle Data Masking and Subsetting. Both tools support comprehensive masking workflows, but they differ in deployment model, feature set, and operational flexibility.

Masking sensitive data using Oracle Data Safe

Oracle Data Safe provides cloud-based static data masking with a rich set of simple masking formats and a library of over 50 predefined formats. To mask a data column, users select the desired masking format from a dropdown list. If Oracle Data Safe's data discovery feature has already been used to identify sensitive data, the tool automatically suggests default masking formats based on the data types it detected. Organizations can also create and add custom masking formats to the library, providing the flexibility to meet unique requirements.

Masking reports

The Oracle Data Safe console summarizes the results of masking operations in detailed reports. Each report contains information about the database and masking policy applied, the timestamp of the masking job, and statistics showing which columns were masked, the masking formats used, and the volume of data transformed. These reports can be downloaded as PDF files to support audit and compliance requirements, helping organizations demonstrate that they are managing data privacy obligations effectively.

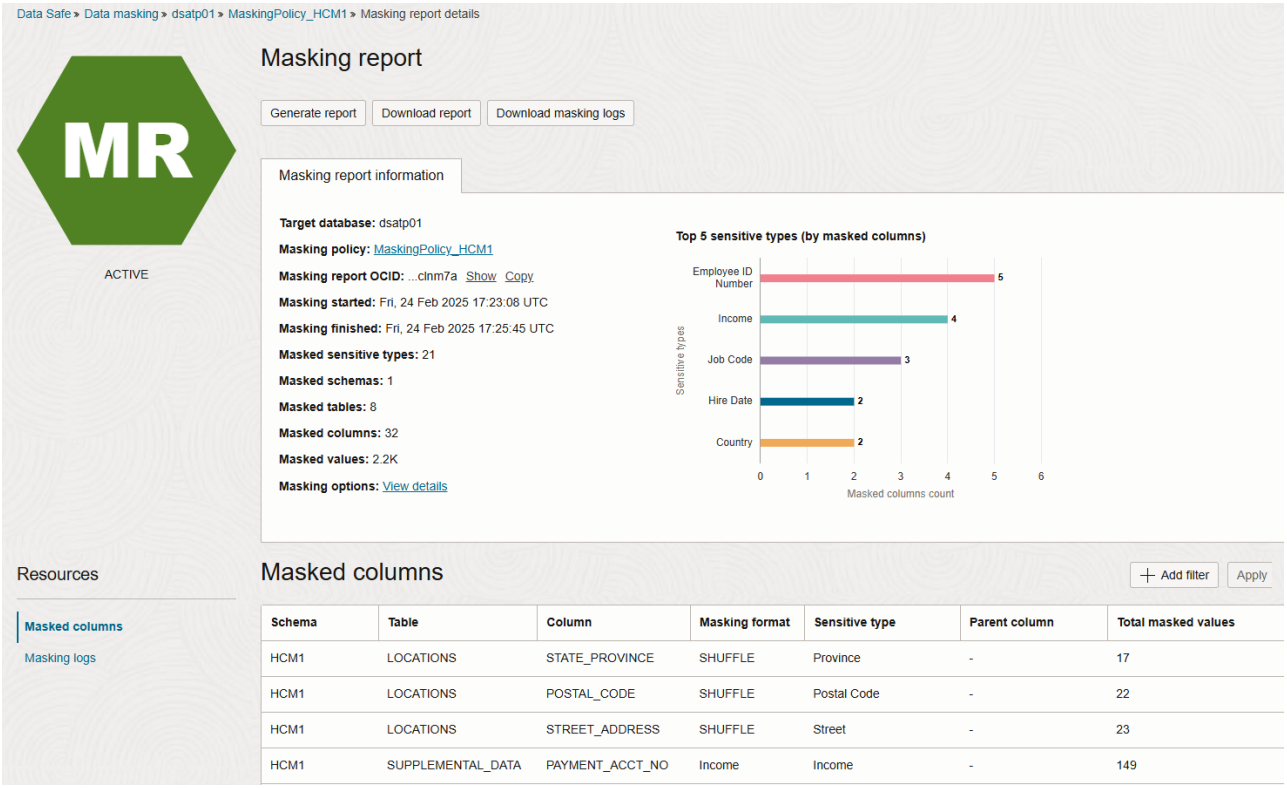


Figure 10-5: Example Data Safe masking report

Masking sensitive data using Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting, available as an Oracle Enterprise Manager management pack, provides static masking and subsetting capabilities for on-premises and cloud deployments.

Data masking workflow

The workflow for Oracle Data Masking and Subsetting is similar to Oracle Data Safe but includes some important differences. Organizations begin by running data discovery to build an application data model that identifies sensitive columns and their referential relationships. Data discovery is a mandatory step with Oracle Data Masking and Subsetting. Only sensitive columns identified through data discovery are eligible for masking.

After sensitive columns have been identified, organizations create a masking definition by assigning masking formats to those columns. Like Oracle Data Safe, Oracle Data Masking and Subsetting supports both simple and predefined masking formats and allows the creation of custom formats.

To apply the masking definition, organizations generate a masking script and execute it within the Enterprise Manager framework or independently. Figure 10-6, taken from the Oracle Data Masking and Subsetting console, provides an overview of the workflow for both masking and subsetting.

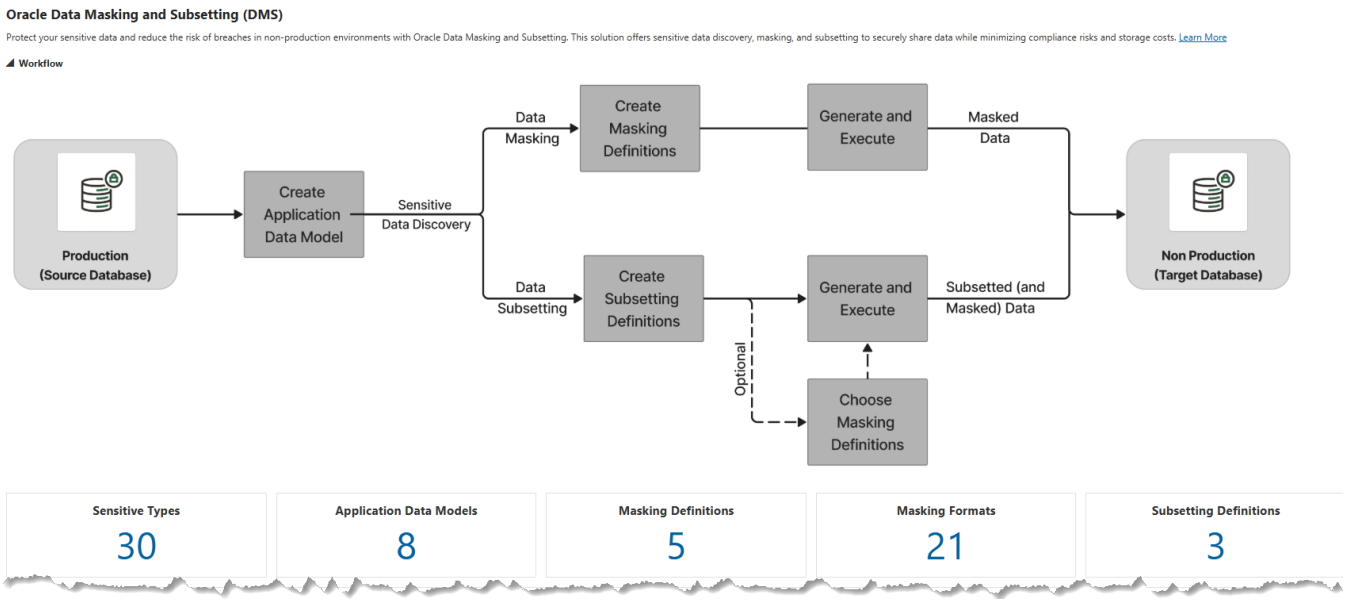


Figure 10-6: Overview of Data Masking in Enterprise Manager

In-database and in-export masking modes

Oracle Data Masking and Subsetting offers two modes for masking: in-database and in-export. The different modes provide flexibility in how masking operations are executed. Figure 10-7 illustrates the difference between the two modes.

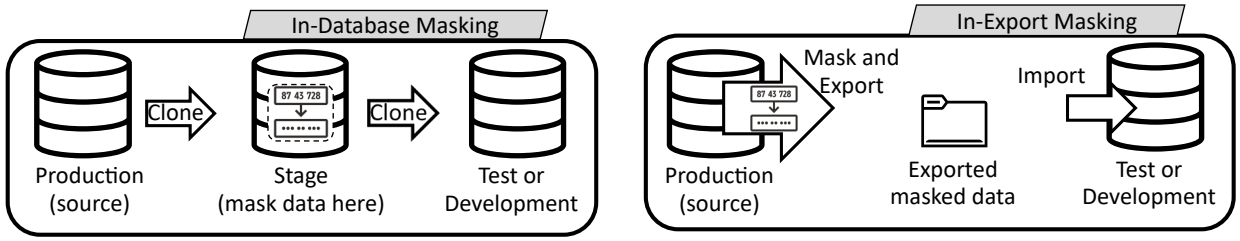


Figure 10-7: Masking modes for Data Masking and Subsetting

The in-database mode applies masking and subsetting within a non-production database, typically a clone of the production database, with minimal impact on the production environment. Organizations start by creating an unmasked staging copy of the production database, then apply masking and subsetting within that staging copy. From this masked staging copy, additional database clones can be created as needed. Data masking irreversibly modifies the stored data and should not normally be performed on a production database.

The in-export masking mode applies masking and subsetting during a Data Pump export operation. In this mode, Oracle Database masks data as it is exported from the production database but leaves the original values untouched. The masked data is written to standard Data Pump files, which can then be imported into non-production databases or shared for analytics. This approach eliminates the need to create a staging copy of the production database and ensures that sensitive data remains only within the production perimeter. When using in-export masking, the masking formats applied should be reasonably simple, and the export process should be scheduled outside peak usage hours to avoid performance impacts.

Choosing between Oracle Data Safe and Oracle Data Masking & Subsetting

Oracle Data Safe and Oracle Data Masking and Subsetting both provide robust static masking capabilities, but they are designed for different deployment scenarios and operational preferences.

Oracle Data Masking and Subsetting works exclusively with Oracle Database Enterprise Edition, while Oracle Data Safe can mask any edition of Oracle Database, including Standard Edition. This makes Oracle Data Safe a more versatile option for organizations with mixed database environments.

Oracle Data Safe streamlines the masking workflow by automatically matching masking formats to sensitive data types identified during data discovery. This automation allows organizations to create and execute masking policies quickly. Oracle Data Masking and Subsetting requires administrators to manually associate each column with the desired masking format, providing greater control but requiring more manual effort.

Oracle Data Masking and Subsetting offers the unique capability to create a masked Data Pump export directly from the production database. While this in-export masking mode limits the complexity of the formats that can be applied, it is a convenient way to generate pre-masked datasets for distribution to remote developers or external analysts.

Oracle Data Masking and Subsetting generates masking scripts that can be downloaded and executed independently of the Enterprise Manager framework. This capability simplifies the orchestration of test and development database provisioning workflows. With Oracle Data Safe, masking execution is always performed by the service and cannot be separated from the platform.

Organizations should evaluate these differences in the context of their specific requirements, including database edition, deployment model, operational complexity, and integration with existing management tools.

Implementing dynamic data masking with Data Redaction

Oracle Database enables dynamic masking of sensitive columns through several mechanisms, including database views, Virtual Private Database (VPD), Real Application Security (RAS), and Data Redaction. Data Redaction, included as part of Oracle Advanced Security, offers the most flexible and powerful approach to dynamic masking.

Data Redaction in Oracle Database

Data Redaction provides one of the most flexible approaches to dynamically masking sensitive data. Rather than altering the data stored in the database, it masks information dynamically as query results are returned. Data Redaction works with the most common column data types and applies across tables, views, and materialized views. Redacted values preserve the key characteristics of the original data, including data type and optional formatting, so downstream applications and processes continue to function as expected.

This seamless behavior extends to both applications and the database itself. Applications receive data in the format they expect, even when sensitive values have been masked. Within the database, redaction leaves buffers, caches, and storage untouched, so it does not interfere with operations such as exporting data with Oracle Data Pump, performing backups and restores with Oracle Recovery Manager, or running advanced configurations such as Oracle Real Application Clusters, Oracle Active Data Guard, or Oracle GoldenGate.

What makes Data Redaction particularly powerful is where and how it operates. Transformation happens inside the database kernel, driven by declarative policy conditions that are transparent and easy to manage. Unlike masking approaches that depend on application code, Data Redaction policies are enforced directly within Oracle Database, ensuring consistent protection across all application modules and tools. This eliminates gaps that can arise when security logic is scattered across multiple codebases or when organizations lack access to application source code.

Oracle AI Database 26ai introduces significant enhancements to Data Redaction, designed for environments where business applications increasingly depend on real-time analytics and complex SQL. Redaction must remain effective even as queries grow more sophisticated. These new capabilities include:

Support for mathematical and set functions. Data Redaction now supports aggregate functions such as SUM, COUNT, MIN, and MAX, as well as set operations such as UNION, INTERSECT, and MINUS. It also extends to advanced SQL constructs such as WITH clauses and OUTER JOINs. Queries execute against the original underlying values in the database, but when query results are returned, only the displayed output is redacted. For example, if a JOIN or UNION includes a redacted column, the operations proceed using the actual values, but the final result is redacted. This change enables complex queries to execute without exposing sensitive data or producing errors.

Support for sorting and grouping on redacted columns. Operations such as GROUP BY, ORDER BY, and DISTINCT can now be applied to columns with redaction policies. The query uses the actual values to group, sort, or remove duplicates, but when the results are displayed, the sensitive column values are redacted. For example, in a hospital setting, patient records may be grouped by insurance policy number. The grouping happens using the actual policy numbers, but the report displays them in redacted form such as XXXXX123.

Views with expressions. Redacted columns can now be used in SQL expressions on both regular and inline views. In addition, functions such as CONCAT, MIN, MAX, COALESCE, and TRIM are supported when applied to redacted columns in views. This ensures consistent redaction across complex view definitions and improves usability in reporting scenarios.

Support for redacting virtual columns in function-based indexes. Virtual columns used in function-based indexes are now automatically redacted when their underlying base columns are redacted. This ensures consistent redaction behavior even when sensitive information is accessed through virtual columns.

Data Redaction is included as part of Oracle Advanced Security. There is nothing additional to install or configure at the database level. Organizations simply create redaction policies and begin protecting sensitive data immediately.

Data Redaction policies

Organizations create Data Redaction policies at the table and column level using the DBMS_REDACT PL/SQL package. Each redaction policy specifies:

- The schema, object, and column to be redacted
- The format of redaction and any parameters associated with that format
- The enforcement expression that determines when redaction is applied

Figure 10-8 shows an example of creating a Data Redaction policy. In this example, the policy specifies that the first five characters of the SSN column in the CUST_INFO table within the HR schema will be redacted whenever the enforcement expression evaluates to true (in this case, always).

```
DBMS_REDACT.ADD_POLICY(
   object_schema       => 'HR',
   object_name         => 'CUST_INFO',
   column_name         => 'SSN',
   policy_name         => 'REDACT_CUST_SSN',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => DBMS_REDACT.REDACT_US_SSN_F5,
   expression          => '1=1')
```

Figure 10-8: Example Data Redaction policy creation for partial redaction

The enforcement expression provides precise control over when to redact data. Organizations can base redaction policies on a wide range of factors, including usernames, client identifiers, database roles, session information such as

client IP addresses and program modules, and context from Oracle Application Express (APEX), Oracle Real Application Security (RAS), or Oracle Label Security (OLS). Multiple conditions can be combined for more granular control.

If organizations use Oracle Enterprise Manager, they can take advantage of the Redaction Policy Expression Builder. This tool guides administrators through creating policy conditions using supported context attributes and provides an intuitive interface for specifying protected columns, transformation types, and enforcement conditions.

Data Redaction formats

Data Redaction policies allow organizations to choose different masking techniques based on business requirements:

- **Full masking**: Masks all sensitive data in a column.
- **Partial masking**: Masks only a portion of the data, such as the first five digits of a Social Security number.
- **Rule-based masking**: Uses regular expressions to define precisely which parts of the data to mask.
- **Random masking**: Replaces sensitive data with random values
- **Nullify**: Replaces sensitive data with a NULL value

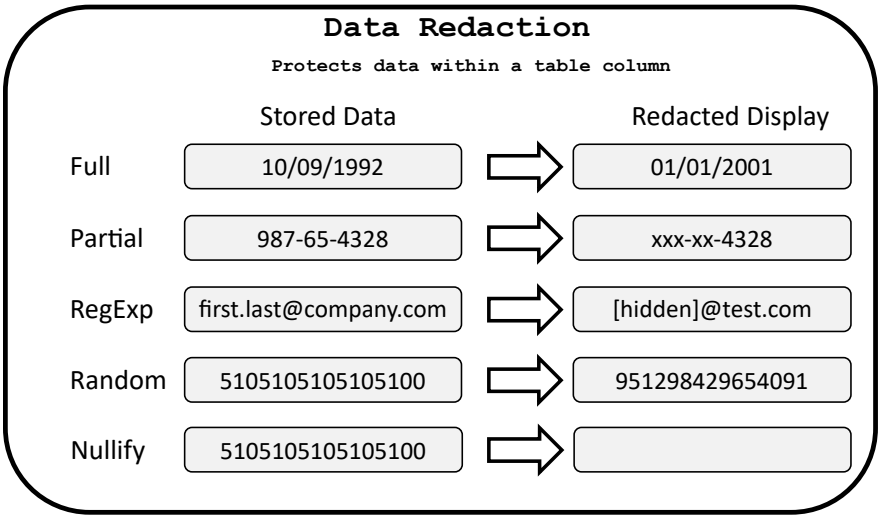


Figure 10-9: Example showing results using various Data Redaction formats

Figure 10-10 and Figure 10-11 demonstrate how application users see data before and after Data Redaction policies are applied.

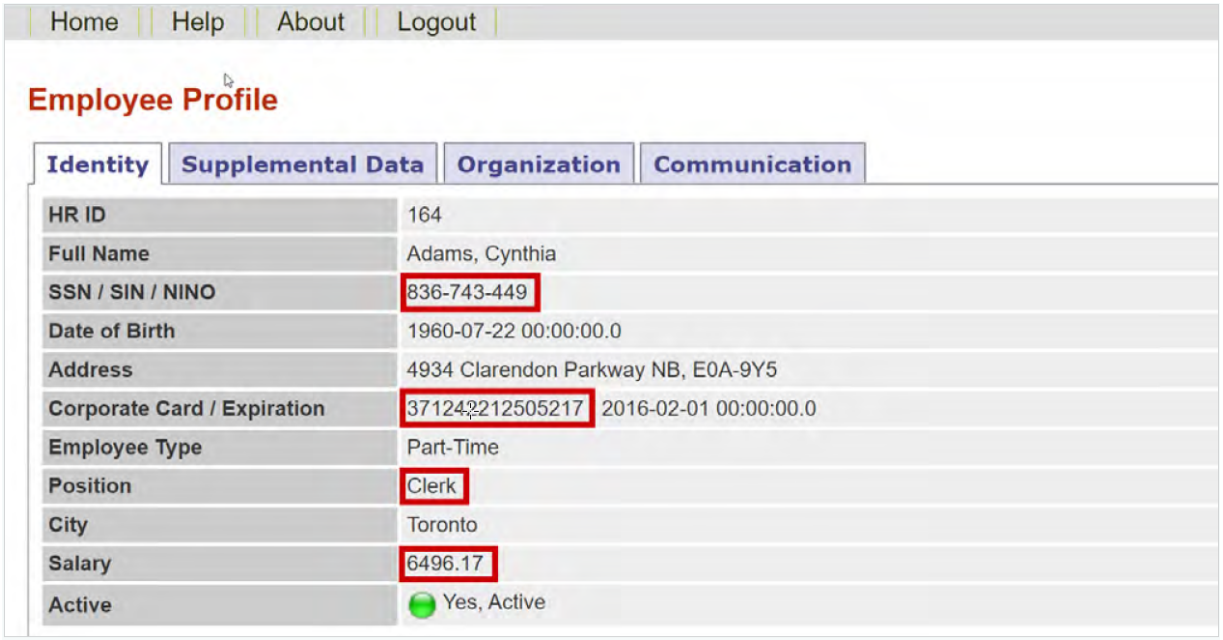


Figure 10-10: Example showing unredacted application data

Figure 10-11 shows the same data with redaction policies applied to the `SSN`, `Corporate Card, Position`, and `Salary` fields using partial, regular expression, random, and full transformations.

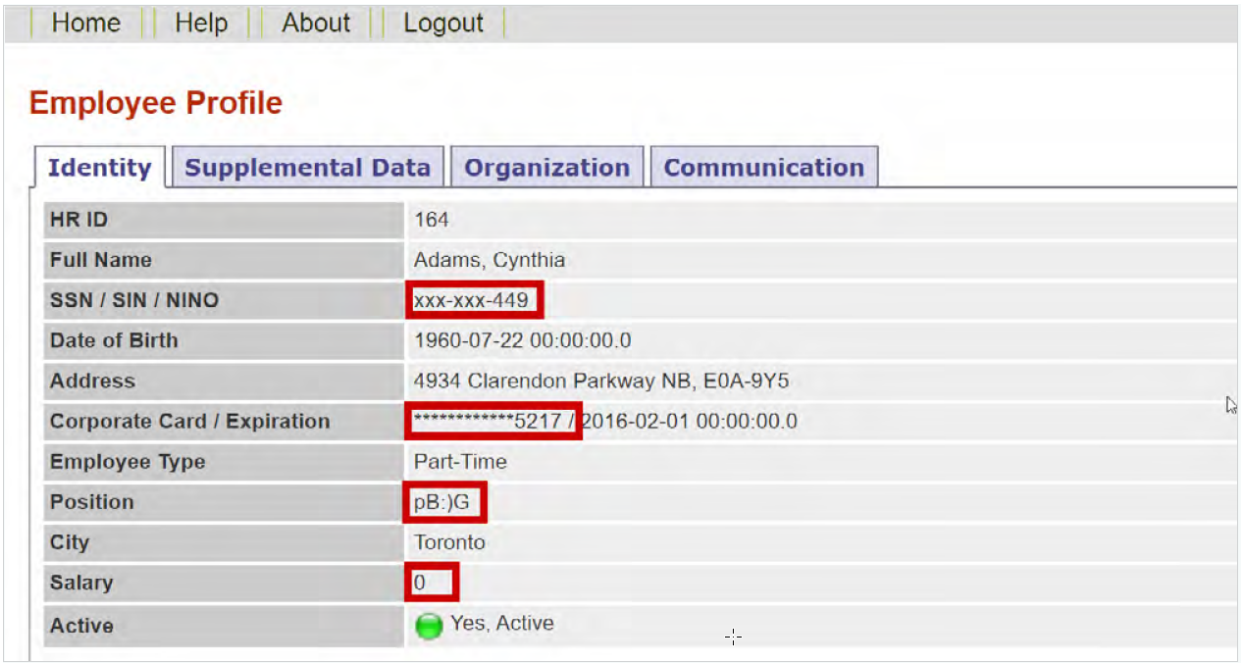


Figure 10-11: Example showing redacted application data

In addition to these simple formats, Data Redaction supports predefined redaction formats for common types of sensitive data, including credit card numbers, Social Security numbers, postal codes, email addresses, and phone numbers. Table 10-3 shows a selection of the predefined redaction formats. Organizations can also create custom redaction formats to match unique requirements.

Format	Description
US SSN first five (formatted)	Redacts the first 5 numbers of a Social Security number when the column is a VARCHAR2 data type. For example, the string 987-65-4320 becomes XXX-XX-4320.
US SSN last four (formatted)	Redacts the last 4 numbers of a Social Security number when the column is a VARCHAR2 data type. For example, the string 987-65-4320 becomes 987-65-XXXX.
US SSN total (formatted)	Redacts the entire Social Security number when the column is a VARCHAR2 data type. For example, the number 987-65-4320 becomes XXX-XX-XXXX.
US SSN first five	Redacts the first 5 numbers of a Social Security number when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXX4320.
US SSN last four	Redacts the last 4 numbers of a Social Security number when the column is a NUMBER data type. For example, the number 987654320 becomes 98765XXXX.
US SSN total	Redacts the entire Social Security number when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXXXXXX.
CA SIN six nines plus last three	Redacts the Canadian Social Insurance number by replacing the first 6 digits by 9 (a number). For example, 123456789 becomes 999999789.
CA SIN last three	Redacts the Canadian Social Insurance number by replacing the first 6 digits by X (string). For example, 123456789 becomes XXXXXX789.
CA SIN last three (formatted)	Redacts the Canadian Social Insurance Number by replacing the first 6 digits by X (string). For example, 123-456-789 becomes XXX-XXX-789.
UK NIN alpha (formatted)	Redacts the UK National Insurance number by replacing the first 6 digits by X (string) but leaving the alphabetic characters as is. For example, ET 27 02 23 D becomes ET XX XX XX D.
UK NIN alpha	Redacts the UK National Insurance number by replacing the first 6 digits by X (string) and leaving the alphabetic characters as is. For example, ET270223D becomes ETXXXXXXD.
Credit Card last four (formatted)	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by *. For example, the credit card number 5105–1051–0510–5100 becomes ****–****–****–5100.
Credit card last four	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by *. For example, the credit card number 5105105105105100 becomes ************5100.
AMEX last five (formatted)	Redacts the American Express credit card number by replacing the digits with * except the last 5 digits. For example, the credit card number 3782-822463-10005 becomes ****-******-10005.
AMEX zeros plus last five	Redacts the American Express Credit Card Number by replacing the digits with 0 except the last 5 digits. For example, the credit card number 3782 822463 10005 becomes 0000 000000 10005.
US Zip code (varchar)	Redacts a 5-digit postal code when the column is a VARCHAR2 data type. For example, 95476 becomes XXXXX.

US Zip code (number)	Redacts a 5-digit postal code when the column is a NUMBER data type. For example, 95476 becomes XXXXX.
Date to 1970	Redacts dates to 01-JAN-1970.
Date to millennium	Redacts dates to 01-JAN-2000.
North America phone number (formatted)	Redacts the North American phone number by leaving the area code, but replacing everything else with X. For example, 650-555–0100 is redacted to 650-XXX-XXXX.
North America phone number (zeros)	Redacts the North American phone number by leaving the area code but replacing everything else with 0. For example, 6505550100 gets redacted to 650000000.
North America phone number	Redacts the North American phone number by leaving the area code, but replacing everything else with X. For example, 6505550100 is redacted to 650XXXXXXX.

Table 10-3: Summary of predefined redaction formats

Implementing Data Subsetting

Oracle Data Masking and Subsetting provides comprehensive data subsetting capabilities that allow organizations to create smaller datasets derived from production data while maintaining referential integrity. For example, a large 2 TB production dataset can be subset down to 100 GB for development or analytics. This smaller dataset enables more efficient operations while using fewer resources.

As shown in Figure 10-12, data subsetting supports several strategies for reducing table size. Organizations can extract a percentage of a large table for application development, pull data from a specific geographic region for localized analysis, or select rows from a particular partition or sub-partition.

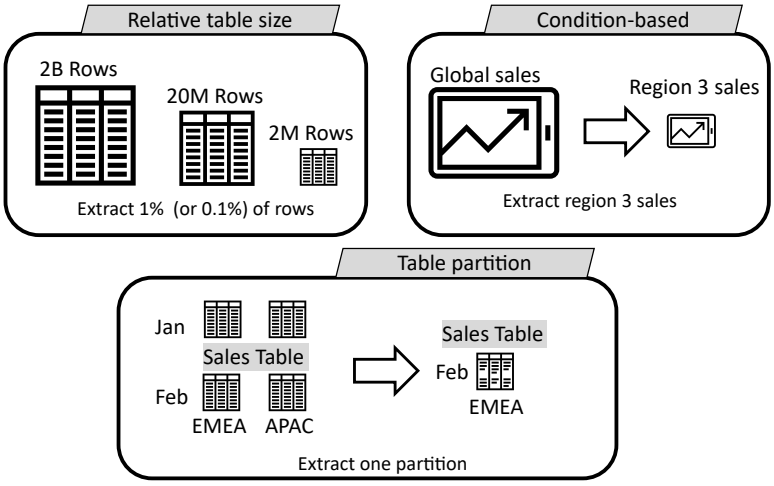


Figure 10-12: Data Subsetting strategies

Table 10-4 describes different methods for structuring the conditions used to extract data for subsetting.

Term	Description
Random Sampling	Randomly selects a subset of data from a dataset. This approach is useful when a sample that represents the entire dataset is needed, such as for development or testing
Stratified Sampling	Use this method when the dataset contains different groups or strata. Stratified sampling ensures that each group is adequately represented, and organizations can define rules for sampling each group.
Time-based Subsetting	Subsets data based on time criteria, such as selecting records from a particular date range. For example, organizations may want the oldest or the most recent data for analytics.
Conditional Subsetting	Selects data based on certain conditions or criteria. For example, organizations might extract records where an attribute meets a certain threshold.
Top-N Queries	Selects a subset of data based on the top or bottom N records for a particular attribute. For example, organizations might extract the top 1000 customers by sales volume.

Table 10-4: Subsetting conditions

Organizations can combine subsetting and masking in a single workflow. First, extract only the data required from production, then mask any sensitive information within that subset before provisioning it to non-production environments. This combined approach maximizes the business value of data, protects sensitive information, and avoids wasting resources on unnecessary storage and processing.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Get Started with Oracle Data Safe Fundamentals
- Data Redaction
- Redaction for Autonomous Databases
- Data Masking and Subsetting

Summary

Modern applications generate and consume huge volumes of sensitive information, from personally identifiable information and financial records to healthcare data and proprietary business details. As that data spreads into development, testing, analytics, training, and daily operations, organizations run into a familiar tension: they want the realism of production data without the exposure that comes from unnecessary visibility.

Realistic datasets unlock real value. Developers validate behavior against believable edge cases, marketing and customer service teams find patterns faster, call centers resolve issues with less back-and-forth, and fraud models learn from meaningful signals. But every new copy of production data widens the attack surface, increasing the odds of misconfigurations, insider misuse, lost devices, and external attacks. The risk climbs even higher when AI and machine learning teams reuse these environments for model training and retrieval augmented generation pipelines, because real PII can end up in places that were never designed to protect it.

Data masking is presented as the pragmatic middle path: convert sensitive data into realistic but fictitious values that preserve utility while stripping away the value of theft. A key point is that masking is not designed to be reversed, and that irreversibility is a feature when sensitive information must be permanently removed from non-production environments. Masking also preserves table relationships through consistent replacements that maintain referential integrity. Alongside masking, the discussion positions data redaction and data subsetting as complementary privacy pillars that support least privilege, anonymization and pseudonymization expectations, and compliance obligations such as PCI DSS, EU GDPR, CCPA, and India DPDPA. A practical example grounds the idea: point-of-sale testing can keep real inventory and store configurations while swapping in plausible names, email addresses, and payment details.

Two primary approaches shape how organizations apply these protections:

- Static **Data Masking** permanently replaces stored values so the target environment contains only masked data. That makes it a strong fit for development, testing, training, analytics, and external data sharing where real customer, employee, or financial details are not needed.
- **Data Redaction**, or dynamic data masking, leaves stored data untouched but transforms query results based on policy. This enables production systems to show different users different versions of sensitive fields without application changes.

Examples reinforce the difference: static masking can replace employee names with realistic substitutes, randomize Social Security numbers into the correct format, and shuffle salaries to preserve totals for aggregate analysis while protecting individual compensation.

Implementation is described as a series of tradeoffs across speed, control, and deployment preferences. For data masking, the chapter compares Oracle Data Safe and Oracle Data Masking and Subsetting: Data Safe emphasizes cloud service execution and automated format suggestions, while Data Masking and Subsetting emphasizes mandatory discovery and more manual format assignment. Masking can also be done in-database using a staging clone, or in-export during Data Pump export. Data Redaction is described as a database kernel capability driven by declarative policies that can incorporate identities, roles, session attributes, and application context, with Oracle AI Database 26ai enhancements to keep redaction effective for complex SQL, grouping, sorting, and views with expressions. Data Subsetting rounds out the toolbox by extracting only the needed slice of production data, cutting risk and cost, speeding refreshes, and pairing naturally with masking in a combined workflow.

The next chapter shifts to protection to encryption, positioning it as a way to keep sensitive data authentic while allowing it to move through production systems with confidence. Continue reading to see how encryption layers onto masking and redaction strategies, and how it can serve as a bright, dependable shield that helps organizations move faster without losing control of what matters most.

Chapter 11

Data Encryption and Key Management

Why encryption matters for database security

Strong authentication and authorization controls protect the front door to the database, but attackers often attempt to bypass these normal access paths. Encryption ensures that data remains unreadable even when attackers circumvent access controls, because encrypted data is useless without the corresponding encryption key. Encryption shifts the security challenge from protecting large volumes of data to protecting a much smaller and more manageable set of keys.

Attackers can bypass database controls in several ways. They may intercept data traveling across the network between database clients and servers; on many networks, capturing traffic and reading clear-text information is straightforward. They can also exploit operating system privileges to read database files directly. Unencrypted backups present another high-value target, whether stored on disk, tape, offsite media, or in cloud storage. Because bypass attacks do not create a database session, traditional access controls offer no protection. Encrypting data at rest and in transit ensures that intercepted traffic, copied files, and stolen media do not expose sensitive information.

Organizations that implement encryption gain several strategic advantages:

- Persistent protection: Data stays protected at rest, in transit, and in backups when encryption is applied appropriately.
- Simpler control surface: Security teams can focus on key security and key lifecycle management instead of tracking every copy of the data.
- Compliance support: Many regulations and security standards require-encryption for sensitive or personally identifiable information (PII). Encrypting data helps organizations meet these requirements and reduce audit findings.

Encryption should be used as part of a defense-in-depth strategy. Organizations must ensure strong key management practices, limit key access according to least privilege principles, and monitor key usage to detect and respond to anomalies quickly. This chapter explains how encryption protects data in motion and at rest for Oracle Database. It also discusses considerations for securely storing and managing the encryption keys that ultimately safeguard the encrypted data.

Encrypting data in motion

Securing data in motion is essential for preventing eavesdropping and tampering. Network encryption is built into Oracle Database across Enterprise and Standard editions, enabling organizations to protect data as it moves between clients and the database. Two primary options are available for encrypting data in motion: industry-standard Transport Layer Security (TLS) and Oracle Native Network Encryption (NNE). Both options provide confidentiality to prevent eavesdropping and integrity protection to prevent tampering or replay attacks. TLS also adds server authentication and can optionally authenticate the client.

Most Oracle client software supports both NNE and TLS, including thin and thick JDBC drivers, Oracle Instant Client, SQL*Plus, SQL Developer, and the Visual Studio Code SQL Developer extension.

Transport Layer Security (TLS)

TLS provides both authentication and encryption for database connections. Organizations can configure server-authenticated TLS, in which the server presents an identifying certificate, or mutual TLS (mTLS), in which both the server and the client present certificates. This approach protects data in motion and can authenticate the client to the database. TLS certificates are stored in the system certificate store or in an Oracle Wallet.

Organizations that use PKI certificate-based authentication for database access depend on TLS to establish the secure channel over which authentication occurs.

TLS cipher negotiation and protocol versions

With TLS, the server and client negotiate and select the strongest mutually supported cipher suite. Organizations can explicitly require specific algorithms if desired.

Oracle AI Database 26ai and newer versions support TLS 1.3, while older versions support TLS 1.2. Organizations whose clients and applications both support TLS 1.3 should use it. TLS 1.3 streamlines the handshake process and uses modern cipher suites, improving security posture and reducing connection setup time compared to TLS 1.2.

Starting with Oracle AI Database 26ai (release 23.26.0), organizations can secure TLS 1.3 connections with Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) or the quantum-resistant ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism) algorithms. This flexibility allows organizations to choose well-established or post-quantum security based on risk profile and compliance needs. The January 2026 release update (23.26.1) added a hybrid key exchange option that combines ECDHE and ML-KEM. In hybrid mode, the database performs key exchange using both algorithms, producing two independent keys that are combined to form the final session key. This dual approach helps ensure that neither classical nor post-quantum vulnerabilities alone compromise the security of encrypted data. As long as at least one of these algorithms remains uncompromised, the encrypted connection is protected.

Native Network Encryption

Native Network Encryption (NNE) encrypts each connection with a unique shared secret key. When a client connects to an NNE-enabled database, the client and server negotiate a session-specific key used only for that connection. Neither side needs a certificate. Database administrators can enable NNE by adding a simple entry to the database network configuration file, and clients require no changes. Because NNE does not rely on external infrastructure or certificates, many organizations choose it for its straightforward setup.

Federal Information Processing Standard (FIPS)

Optionally, TLS and NNE can run in FIPS mode to enforce stricter control over approved cryptographic libraries. Oracle AI Database 26ai allows organizations to choose between FIPS 140-2 and FIPS 140-3 modes.

Security and performance considerations

Stronger ciphers typically increase compute overhead. However, TLS 1.3 often performs better than TLS 1.2. The updated handshake removes an extra round trip, which speeds up connection establishment, and the standardized cipher suites are modern and efficient, delivering slightly better performance while strengthening security.

For more information on configuring TLS encryption, refer to the Oracle AI Database Security Guide.

Encrypting data at rest

Organizations can encrypt data at multiple points in the technology stack: the application layer, the database layer, or the storage layer. In general, the lower encryption is implemented in the stack, the less intrusive and better performing it is. However, encrypting lower in the stack addresses fewer threat types.

Encryption strategies: application, database, and storage layers

Application-layer encryption sits at the top of the stack. Applications encrypt data before writing it to the database and decrypt it when needed. Each application must manage encryption and decryption throughout its workflows. This approach can affect database performance and reduce the types of queries that can run on encrypted columns. For example, common analytic queries that search ranges or compute values may be slow or fail because indexes are often less useful with application-encrypted data. If multiple applications share the same database, they must also share access to the same encryption keys to encrypt and decrypt data.

Application-layer encryption provides strong protection but adds significant development and operational overhead, and it limits meaningful relational operations on the data. Because the database may no longer use indexes effectively, performance can degrade. Application-layer encryption can make sense for narrow cases, such as storing a secret string or a blob that must not be visible to any other database user, including administrators, and that does not require database operations.

Database-layer encryption encrypts whatever data the application sends and stores the result in encrypted form. Multiple applications can use the same database without changes because the encryption is transparent to them. This protects against attacks that bypass the database and attempt to read data directly from storage. It does not protect against someone who connects through the database with authorization to read the data. Access controls remain critical: access controls guard the front door; encryption protects the windows and back doors.

Storage-layer encryption (file or volume encryption) protects data at rest against someone who bypasses the operating system and accesses storage directly. This provides limited mitigation for database threats. The main problem is that file or volume encryption lacks database user context. If Oracle Database runs as the operating system user "oracle," any other user logged in as "oracle" or "root" can access decrypted data. Because the goal is to prevent database bypass, a control that automatically decrypts data for operating system utilities that read or copy files (for example, file-search and file-copy tools) is not effective.

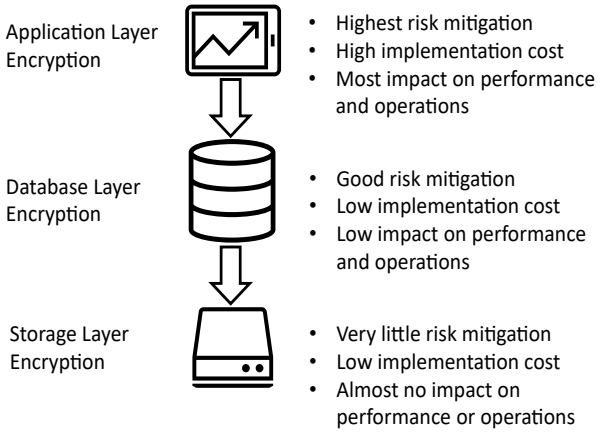


Figure 11-1: Encryption offers trade-offs at each layer

Storage-layer encryption usually has minimal impact on stability and operations, but results depend on the solution. Encrypting file systems such as ZFS or Microsoft BitLocker, or secure-wipe encryption built into high-end media such as that used in Exadata storage systems, tend to be stable. Third-party file system encryption that inserts itself between the operating system and the database can introduce instability and complicate upgrades. It can also delay patching because teams must wait for the file system vendor's dependent patch before applying database or operating system patches. Because of these complications, Oracle cannot support database issues that arise from the use of third-party file system encryption.

Evaluating encryption solutions

Regardless of where-encryption is implemented, organizations should evaluate solutions by the following criteria:

- Which risks does the encryption mitigate?
- What is the performance impact?
- How simple is installation and use?
- How transparent is it to existing applications?
- What resources are needed to convert data between plaintext and encrypted forms?

- What are the patching considerations?
- How robust is key management?

Neither application-layer nor storage-layer encryption alone is optimal for protecting data in Oracle Database. Organizations need a mechanism that improves security and reliability while minimizing implementation effort and performance overhead.

Combining encryption strategies

One size rarely fits all. Many organizations blend multiple encryption strategies. For example, a retailer that retains financial data, including customers' credit cards for reuse, and personal data such as date of birth, physical address, email address, phone number, and purchase history might implement:

1. **Application-layer encryption for credit card numbers**: Data is encrypted at the point of collection and remains encrypted except when used to support purchases.
2. **Database-layer encryption for all other application data:** This protects against bypass attacks such as ransomware and limits data theft exposure while allowing analytics and, where customers have opted in, marketing use.
3. **Storage-layer encryption**: The SAN uses media configured for secure wipe when failed hardware must return to the manufacturer.

Transparent Data Encryption

When attackers try to bypass database controls and read files directly, data must remain protected. Transparent Data Encryption (TDE), a feature of Oracle Advanced Security, encrypts data within the database to ensure it remains unreadable outside an authenticated session.

How TDE works

TDE encrypts data before writing it to storage and decrypts it when reading it back. Authorized users and applications see decrypted (plaintext) data, but anyone who tries to read data through the operating system or from storage sees only encrypted data. Because database files are encrypted, backups that include that data are also encrypted, so stolen backup copies do not expose sensitive information.

Nothing changes for applications. Organizations keep using the database as they do today. The database enforces the access controls that have been defined and denies access to anyone who is not authorized, while TDE protects data at rest from bypass attacks.

This transparency is equally valuable for emerging AI workloads. Data derived from sensitive sources, including vector embeddings, training datasets, and inference caches, deserves the same protection as the original production data, and TDE provides this protection at the tablespace level with no application modifications.

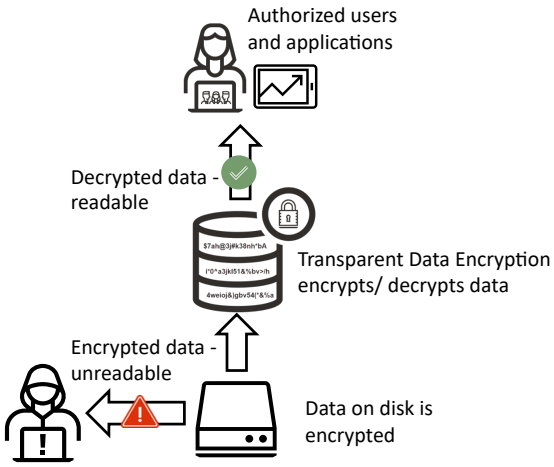


Figure 11-2: Users and applications access the decrypted data in database memory

Tablespace encryption and column encryption

TDE can encrypt whole tablespaces or individual columns. In almost all cases, tablespace encryption will be the optimal choice. With tablespace encryption, database administrators do not need to track which columns to encrypt or worry about someone inserting sensitive data into a column that would not usually be considered sensitive. With tablespace encryption, administrators also do not need to worry about column characteristics such as indexes and constraints.

Creating encrypted tablespaces

Creating an encrypted tablespace is straightforward:

```
SQL> CREATE TABLESPACE investigations
     DATAFILE '$ORACLE_HOME/dbs/investigations.dbf' SIZE 50M
     ENCRYPTION [USING 'AES256' MODE 'XTS'] ENCRYPT;
```

Figure 11-3: Example syntax creating an encrypted tablespace

Note: The default encryption algorithm in Oracle AI Database 26ai is AES256 using XEX-based tweaked-code book mode with ciphertext stealing (XTS) encryption. Shorter key lengths (AES192 and AES128) are available, but not advised due to concerns about quantum computing resistance. Pre-26ai databases implemented AES using cipher feedback (CFB) mode. Oracle AI Database 26ai still offers CFB as an option for backward compatibility.

Organizations may configure their databases to encrypt new tablespaces automatically, even if an application does not create encrypted tablespaces as part of the application installation process.

When TDE is used to encrypt tablespaces, sensitive data remains encrypted throughout the database, whether in tablespace storage files, temporary or undo tablespaces, or other files such as redo logs. In addition, TDE can encrypt RMAN database backups and Data Pump exports.

TDE automatically leverages special instructions in Intel/AMD (AES-NI), SPARC, ARM, and IBM Power CPUs to accelerate cryptographic operations. In addition, TDE tablespace encryption integrates with the performance optimizations built into Oracle Engineered Systems such as Exadata and the Oracle Database Appliance. For example, TDE tablespace encryption works seamlessly with Exadata Hybrid Columnar Compression (EHCC) and Smart Scan technology.

Migrating existing data to encrypted tablespaces

When encrypting tablespaces in an existing database, there are two choices – online and offline.

Online tablespace encryption, introduced with Oracle Database 12.2.0.1 enables zero-downtime migration from plaintext to encrypted data or converts from one encryption algorithm to another. Online encryption temporarily uses additional storage of the same size as the largest processed tablespace. In the sample command shown in Figure 11-4, datafile hr_data01.dbf will be copied and written in encrypted form into datafile hr_data01_enc.dbf. When the encryption operation completes, the original datafile hr_data01.dbf will be overwritten with zeroes and deleted.

```
ALTER TABLESPACE hr_data ENCRYPTION ONLINE [using ‘algorithm’] ENCRYPT
[FILE_NAME_CONVERT=('hr_data01.dbf','hr_data01_enc.dbf')];
```

Figure 11-4: Example syntax for online tablespace encryption

TDE also supports two offline encryption methods that do not require extra storage but do require the tablespace to either be offline or the database to be in mount mode. Both are "standby-first" migration methods if the database is protected by a standby database. The procedure is to turn off the managed recovery process, encrypt the standby database, resume managed recovery and wait for the recovery process to catch up, then perform a switchover and encrypt the new standby (formerly primary) database. In this scenario, the downtime for encrypting any size database is as short as the duration of one or two switchovers.

The first offline encryption method simply has administrators take the tablespace offline and issue an ALTER TABLESPACE command to encrypt all data files in the tablespace. The command can be as straightforward as shown below if the default encryption algorithm and mode are acceptable (AES256 in XTS mode in Oracle AI Database 26ai, AES128 in CFB mode in Oracle Database 19c). If desired, administrators can override the defaults by specifying the algorithm and mode of choice.

```
ALTER TABLESPACE hr_data ENCRYPTION OFFLINE [using 'algorithm' mode 'XTS'] ENCRYPT;
```

Figure 11-5: Example syntax for offline tablespace encryption

The second offline encryption method allows administrators to encrypt the data files, potentially multiple in parallel:

```
ALTER DATABASE datafile '$ORACLE_HOME/dbs/investigations.dbf' ENCRYPT;
```

Figure 11-6: Example syntax for offline data file encryption

This command encrypts the data files with the default encryption algorithm.

TDE and the Database Configuration Assistant (DBCA)

Starting with Oracle Database 21c, DBCA can create encrypted databases with a wallet-based TDE setup, so the database is encrypted from the very start. In addition to creating a new encrypted database, DBCA in 26ai allows organizations to:

- Create new encrypted databases with their keys managed in Oracle Key Vault (OKV).
- Encrypt an unencrypted database.
- Migrate an encrypted database from wallet-based TDE setup to centralized key management with OKV.

Key management

Encryption is only as strong as key management. Organizations must store keys securely, restrict access according to least privilege principles, and validate backup and recovery processes. This section describes options for key management, beginning with local key management using Oracle Wallets and progressing to centralized key management with Oracle Key Vault.

TDE two-tier key architecture

At the start of this chapter, the message was clear: "*Encryption reduces the problem of securing a large amount of data to a more straightforward problem of securing a much smaller key used to encrypt the data.*" That idea sits at the heart of encryption in the real-world. Encrypted data stays protected only as long as the encryption keys stay secret, which is why disciplined key management is so important.

With the right key management approach, attackers are far less likely to discover an encryption key and unlock access to sensitive data. It also helps keep active keys from being misplaced, ensures keys are rotated periodically, and makes sure older keys are securely archived so encrypted backup data sets remain usable over time. Many regulations, including PCI-DSS, require physical separation between encrypted data and encryption keys and recommend periodic key rotation to shrink the exposure period if a key is ever compromised.

TDE uses a two-tier key architecture to create and manage the keys used for encryption. The first tier is a master encryption key (MEK) used throughout the database. The second tier is a set of data encrypting keys (DEK) that are unique for tablespaces or tables, depending on whether column or tablespace encryption is used.

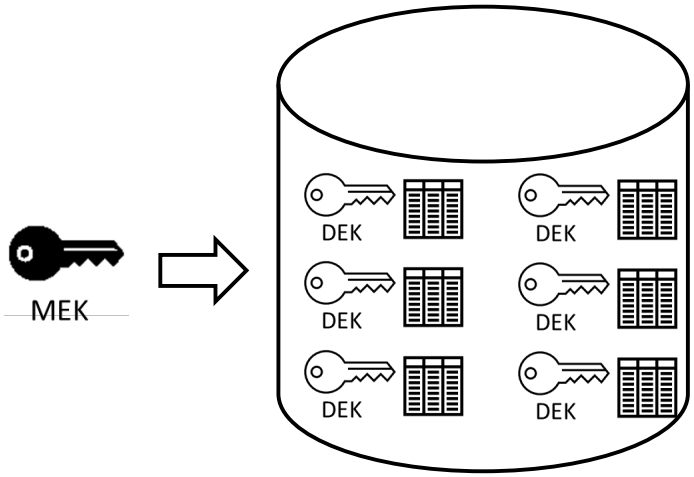


Figure 11-7: TDE uses a two-tier key architecture to improve key management security

As shown in Figure 11-7, a single MEK is kept outside of the database and is used to encrypt and decrypt the internally generated DEK. Those DEKs, in turn, encrypt tablespace data files or table columns. The database automatically manages DEKs behind the scenes.

This two-tier key architecture streamlines key rotation. When rotating the MEK, there is no need to decrypt and re-encrypt all data, only the much smaller set of DEKs. Online tablespace encryption can be used if there is ever a need to rotate the DEKs or switch to another encryption algorithm.

Administrators perform key management operations, such as create, open, rotate, and backup, through SQL commands or, if the database is an Oracle Cloud database service, through the OCI console. Oracle Database 12c introduced the SYSKM administrative privilege, enabling key administrators to carry out key management operations without requiring the powerful SYSDBA privilege.

The MEK is always stored outside of the database, separated from encrypted data, and managed by the key administrator in the Oracle Wallet, Oracle Key Vault, or OCI Vault as shown below in Figure 11-8.

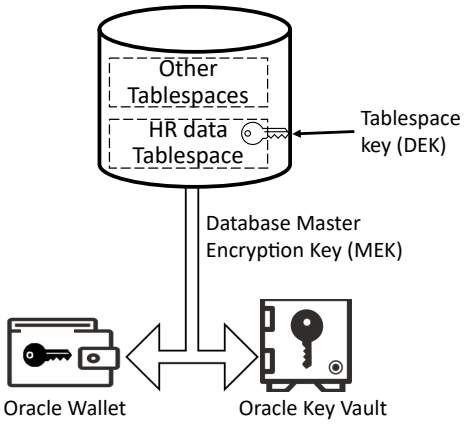


Figure 11-8: Example TDE encryption keys managed by either Oracle Wallet or Oracle Key Vault

The database master encryption key is created when TDE is first enabled, and it can be rotated using an `ADMINISTER KEY MANAGEMENT` command. Tablespace keys are automatically created when a tablespace is encrypted, and a tablespace key can be rotated using an `ALTER TABLESPACE` command. Each encrypted tablespace has its own tablespace key, which is stored in the tablespace header and encrypted using the master key.

With Oracle AI Database 18c and later, externally generated MEKs can be imported into the key store, enabling bring your own key (BYOK) functionality for those who prefer to use an external key generator.

Local key management with Oracle Wallets

By default, Transparent Data Encryption (TDE) stores the master encryption key in an Oracle Wallet that follows the PKCS#12 standard. The wallet's contents are encrypted with an AES256 key derived from the wallet passphrase. If the wallet is lost or corrupted, or the passphrase is forgotten, the encrypted data cannot be recovered. TDE has no built-in recovery 'back doors' for lost passwords. Organizations must securely back up the password-protected wallet and test recovery regularly.

To streamline daily operations, organizations can use local auto-login wallets so the database can open the wallet without manual passphrase entry. A local auto-login wallet is tied to the server where it is created, which reduces the risk of misuse on another system. Some operational guidance when using wallets includes:

- Limit wallet and key access to authorized administrators only.
- Protect wallet files with strong operating system permissions and encrypted backups.
- Document and test wallet backup and recovery steps on a defined schedule.
- Rotate keys according to policy and coordinate rotations with maintenance windows.
- Monitor key usage and alert on anomalies.

Centralized key management with Oracle Key Vault

As more databases are encrypted, managing hundreds or thousands of wallets becomes risky and time-consuming. Lost or corrupted wallets and forgotten passphrases can put data and entire databases at risk. Many regulations and security best practices also prohibit storing encryption keys on the same server as the encrypted data, and many organizations require clear separation of duties between database administrators and key management administrators.

Oracle Key Vault (OKV) helps organizations solve these challenges. With OKV, organizations centralize and secure master keys, wallets, and other secrets, reduce the operational burden of wallet sprawl, and lower the risk of losing access to encrypted data. Key Vault provides:

- **Central control**: Store, manage, and distribute TDE master keys, wallets, and secrets from a single secure service.
- **Strong security**: Keep keys off database servers and enforce least-privilege access with separation of duties.
- **Operational resilience**: Automate backup of wallets and keys, streamline rotations, and reduce the chance of lockouts.
- **Compliance support**: Align with regulations and security best practices that require externalized key management and auditable controls.

With Key Vault, organizations gain an enterprise-grade key manager that secures and reliably distributes the keys and secrets their databases depend on, while simplifying day-to-day operations.

Oracle AI Database connects to the Key Vault (or, more typically, the Key Vault cluster) through a PKCS#11 library installed on the database server as shown in Figure 11-9 below.

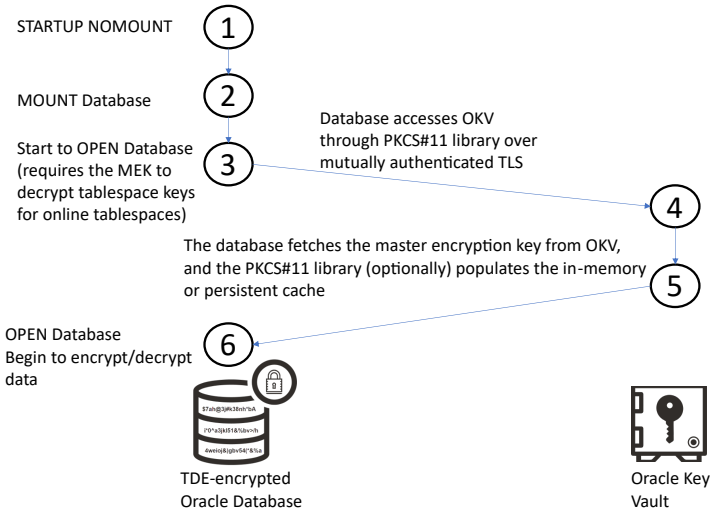


Figure 11-9: Oracle Key Vault securely delivers MEKs to Oracle AI Databases on demand.

Oracle Database connects to the Key Vault (or, more typically, the Key Vault cluster) through a PKCS#11 library installed on the database server. The master key is used for many database operations. Operations that require the database to consult the MEK include:

- When the database is started (opened)
- When a new tablespace is created
- When a tablespace or data file is brought online
- When a redo log group switch occurs
- When a new archive log is created
- When a parallel query process is begun
- During direct-path access to the database
- Initial access to an encrypted column

- GoldenGate and Data Guard operations

There is also a "heartbeat" check of the external keystore to ensure it remains available. If the database cannot access the master encryption key, it immediately closes the keystore to preserve data and prevent corruption. To avoid downtime, organizations should keep Key Vault (OKV) highly available so it can respond to key requests quickly. OKV delivers continuous availability and high scalability for read and write key operations using a multi-master cluster architecture. Organizations can create clusters with up to 16 nodes across geographically distributed data centers. As long as the database can reach any node in the cluster, operations continue. Organizations can place nodes on-premises and in major cloud providers (Oracle, Amazon, Microsoft, and Google). Because all nodes are active, each node holds a copy of all keys and shares the key management and distribution load.

Databases can connect to any OKV node to retrieve master encryption keys. OKV synchronously replicates key updates and authorization changes to a partner node to protect against data loss if a node fails. If a connection fails or a node goes down, database servers transparently fail over to a nearby active node for read and write operations without downtime or administrator intervention.

To guard against transient network issues, the default configuration uses a local persistent cache. OKV endpoints cache keys in an encrypted software keystore on the database server and serve them if the network link to OKV is unavailable. Organizations can protect the persistent cache with a random password that differs from the keystore password. Most key operations are satisfied from this cache, which improves performance and reduces dependency on the network.

Organizations can configure keys as non-extractable so they never leave the OKV cluster boundary. When the database must decrypt a data encryption key (DEK), it requests the decrypted DEK over an encrypted channel from OKV.

Multi-cloud and hybrid use cases

Oracle Key Vault allows organizations to retain ownership and portability of encryption keys, enabling seamless database movement across OCI, Azure, AWS, Google Cloud, and on-premises deployments. This empowers enterprises to meet regulatory requirements for key control while ensuring operational flexibility across diverse infrastructure environments.

Unlike native cloud KMS solutions, which lock encryption keys to a single cloud provider or environment, Oracle Key Vault enables several strategic use cases:

- **Multi-cloud disaster recovery**: Architecting a primary database in one cloud (for example, AWS) with a standby in another (for example, Azure) is unworkable with native KMS, as keys cannot be accessed cross-cloud. OKV provides an independent key management fabric, allowing both sites to operate and failover seamlessly.
- **Hybrid and multi-cloud operations**: Relying solely on native KMS creates a fragmented security posture, requiring management of multiple incompatible key management systems across clouds or between on-premises and cloud. OKV consolidates key management for all environments into one consistent platform.
- **Backup portability and repatriation**: Database backups are tied to the KMS that encrypted them, making cross-cloud or on-premises restoration impossible with native KMS. OKV ensures keys are independent and accessible from any environment, enabling true backup portability.
- **Intra-cloud regional migration**: Even within a single cloud, migrating databases between regions can result in dependencies and risk if keys remain in the original region's KMS. OKV's distributed architecture provides local key access in every location, ensuring regional independence and resilience.

- **On-premises clean room recovery**: Security best practices often require restoring cloud backups to isolated on-premises environments for forensics or breach recovery. Native KMS cannot support this, but OKV can be deployed within the clean room, enabling secure recovery.
- **Multi-cloud sharding**: For globally distributed sharded databases that span multiple clouds, native KMS solutions create "encryption islands." OKV's cross-cloud key management allows sharded architectures to function as a single logical database.

By delivering these capabilities, Oracle Key Vault unifies and simplifies key management to support modern data mobility, disaster recovery, and security requirements across hybrid and multi-cloud environments.

OKV supports TDE keys across Oracle Database deployments that use Multitenant, Oracle Real Application Clusters (RAC), Data Guard, globally distributed (sharded) databases, GoldenGate, and other Oracle technologies. Beyond TDE, OKV provides SSH key governance and manages:

- Java Keystores
- MySQL and MongoDB encryption keys
- Solaris Crypto keys
- Oracle ASM Cluster File System (ACFS) volume encryption keys

If organizations prefer local wallets, OKV can back them up on a schedule as a key escrow service. OKV can also back up Java Keystores and PKI certificate wallets.

Key Vault with DBMS_CRYPTO

Separating keys and secrets from application code increases security and simplifies development. Key Vault offers an integration accelerator that allows developers, after adapting a few configuration files to their environment, to implement key management for custom cryptographic applications using Oracle Database's DBMS_CRYPTO package.

Key Vault for SSH key governance and secrets management

Beyond encryption, Oracle Key Vault (OKV) helps organizations govern SSH keys and control remote server access across their environment. Centralized SSH key management protects private and public keys from loss and misuse and allows organizations to define how keys are created, distributed, rotated, and revoked. During a security incident, organizations can disable SSH access across servers with a single action or API call. Because OKV is the central point for SSH key access, it also provides a clear audit trail that shows who accessed a server, when access occurred, and the source of the request.

OKV is also a comprehensive secrets manager for credentials used in daily IT tasks. Many teams rely on locally managed Secure External Password Store (SEPS) wallets for jobs such as nightly RMAN backups, batch data loads, or refreshing materialized views. Because those wallets reside on the database server, they can be stolen. Instead, organizations can store database account passwords in OKV and retrieve them securely at runtime via an API call. Centralizing credentials in OKV strengthens security and removes the overhead of protecting hundreds or thousands of SEPS wallets used for automation. It also simplifies tightly controlled sharing of those passwords between databases.

C and Java SDKs let developers integrate Java, C, or C++ applications with OKV to manage encryption keys, passwords, and other secrets and retrieve them on demand.

OKV protects both public keys and non-extractable private SSH keys from loss and misuse. With centralized SSH key governance, organizations can enforce consistent policies fleet-wide and, when needed, shut off SSH access quickly during an incident while retaining a complete audit trail.

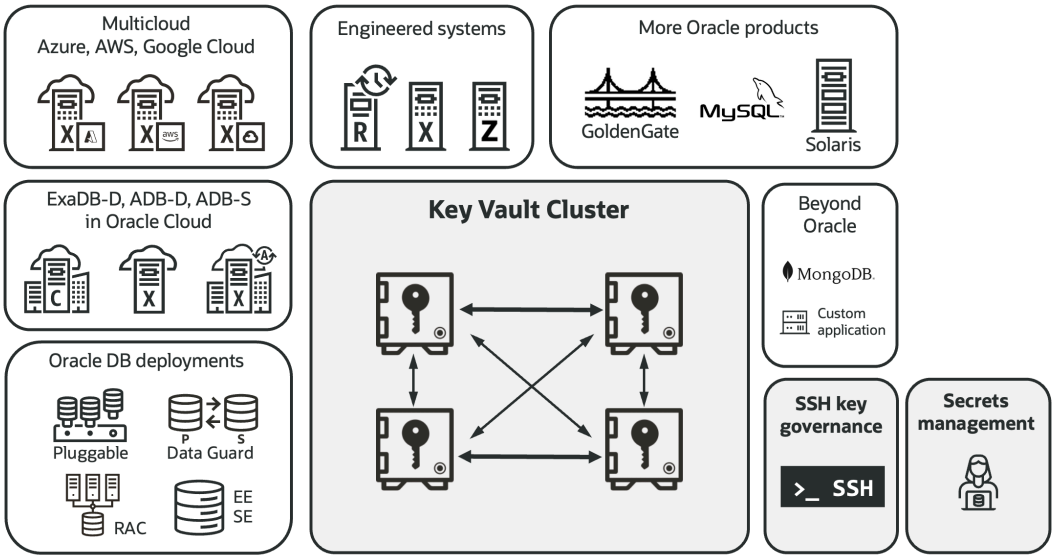


Figure 11-10: Oracle Key Vault manages Oracle TDE keys as well as other keys and security objects.

Key Vault APIs for automation

Key Vault's RESTful APIs make it easy to monitor a Key Vault cluster, add or remove nodes from an existing cluster, or even create a new cluster. Onboarding new servers and managing encryption keys may also be automated through the APIs, with the process of distributing and enrolling Key Vault endpoints fully automated. Automation allows database administrators to encrypt their databases while the security team securely manages the master keys within Oracle Key Vault.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- DB Security – Native Network Encryption
- Protect database communications using 1-way TLS
- DB Security – ASO (Transparent Data Encryption)
- Pluggables, Clones, and Containers – Securing Oracle Multitenant with TDE
- DB Security – Key Vault
- DB Security – Key Vault (SSH Key Management)

Summary

Encryption is the safety net that keeps working even when other defenses get sidestepped, especially in bypass scenarios where no database session is ever created. Real-world attack paths make the stakes concrete: sniffing data on the wire, using operating system privileges to read database files, or simply walking off with unencrypted backups sitting on disk, tape, offsite media, or in cloud storage. In practical terms, encryption turns exposed data into gibberish unless the attacker also gets the key. That shift naturally moves the spotlight to strong key custody, disciplined key lifecycle practices, and sharp-eyed detection of suspicious key activity.

On the network side, protecting data in motion is best handled with TLS or Oracle Native Network Encryption. Both provide confidentiality and integrity protections that help block tampering and replay. TLS stands out as the more identity-rich option because it authenticates the server and can also authenticate the client through mutual TLS, using certificates stored in an Oracle Wallet or the system certificate store. Version and algorithm choices matter too: Oracle AI Database 26ai and newer support TLS 1.3, which improves handshake efficiency compared to TLS 1.2, with additional options such as ECDHE, ML KEM, and a hybrid exchange combining the two. Oracle Native Network Encryption, by contrast, is often the smoother operational path. It avoids certificates, can frequently be enabled with straightforward network configuration, and still generates a unique key for each connection.

For data at rest, it helps to think in layers: application layer, database layer, and storage layer encryption. Lower layers can be less intrusive, but they may cover fewer threat types. Application layer encryption can raise the cost of development and operations, and it can restrict the database's ability to use indexes and run certain analytic queries. Storage layer encryption can also come up short against bypass attacks when operating system users like `oracle` or `root` can access decrypted data. For Oracle Database, Transparent Data Encryption (TDE) is the purpose-built database layer choice: it encrypts data before it is written to storage, keeps it unreadable outside an authenticated session, and encrypts backups that include encrypted files, all without requiring application changes. In almost all cases, tablespace encryption is the recommended approach over column encryption. It's also important to plan how encryption gets applied: encrypted tablespaces can be created directly, and existing data can be encrypted through online conversion or offline methods, including standby-first patterns that keep downtime largely limited to switchovers.

The unifying theme is simple and unavoidable: encryption is only as strong as the keys behind it. TDE's two-tier key architecture helps operationally because an externally stored master encryption key protects internal data encrypting keys, and rotating the master key does not require decrypting and re-encrypting all data. Key storage, however, comes with real operational risk. Wallet-based storage using PKCS#12 Oracle Wallets works, but losing the wallet or forgetting the passphrase can make encrypted data unrecoverable. That's why disciplined backups and regular recovery testing are essential. Auto-login wallets can reduce day-to-day friction, but they're tied to a single server. At enterprise scale, Oracle Key Vault offers centralized key management that reduces wallet sprawl, supports separation of duties, provides high availability via multi-master clustering, adds local persistent caching to reduce network sensitivity, and can keep keys within the cluster boundary through non-extractable configuration while still meeting database needs over secure channels.

The next chapter turns from locking data down to lighting the environment up. Database Auditing and Database Activity Monitoring focuses on seeing what is happening as it happens, where the database tells its story in crisp, actionable signals that help teams validate controls, spot anomalies, and transform security from a hope into a measured, watchful advantage.

Chapter 12

Using and Managing Audit Data

Why auditing matters

Auditing is often viewed as one of the less exciting areas of security, something everyone knows they should do but too often postpones. There are many reasons for this. Some don't know what to audit; some don't know how to audit; some don't know what to do with the audit records once they have audited; and some believe the audit performance overhead is a penalty that doesn't justify the process.

Simply collecting audit records is only the first part of the equation. That collected data has limited value unless it is reviewed and acted on. Database activity should be actively reviewed and monitored because accounts remain at risk of being compromised or misused, and early detection can help reduce the damage from a compromise.

As AI agents and Model Context Protocol (MCP) pipelines connect to databases alongside human users, an audit strategy must account for both. Machine-generated queries can arrive at high volume, at unusual hours, and through service accounts that can blur traditional accountability. Capturing and distinguishing this activity is essential to maintaining visibility and control.

The value of database auditing

Database auditing helps organizations catch malicious behavior sooner, satisfy regulatory obligations, and move faster during incident investigations. It also creates an authoritative audit trail that shows who did what, when it happened, and where the request came from, bringing clarity, accountability, and stronger security. The data collected through auditing is useful for all of the following:

- **Detecting and understanding malicious activity**. Spot suspicious behavior quickly, uncover root causes, and reduce dwell time.
- **Assisting investigations**. After a breach or other suspicious activity, replay the timeline to see exactly what happened and when.
- **Deterring inappropriate behavior**. When people know activity is audited, they are less likely to take risky or malicious actions.
- **Non-repudiation**. Provide proof that an event occurred, including who performed it, when it happened, and the source of the request.
- **Supporting regulatory compliance audits**. Show a verifiable record of activity, especially access to or changes in financial and other sensitive data.
- **Monitoring privileged user activity**. Keep visibility into actions by database administrators (DBAs) and other privileged accounts that access critical systems and data. These high-value accounts are frequent targets for attackers.
- **Assisting troubleshooting.** For example, if hundreds of application servers connect to a database and one keeps trying to log in with the wrong credentials, failed login auditing helps you find the culprit and fix the issue quickly.

When monitoring and auditing become a standard part of day-to-day operations, organizations reduce risk, strengthen compliance, and give their teams the evidence they need to act quickly and accurately.

Oracle Database unified auditing

Oracle Database auditing is designed to be both robust and highly customizable, so organizations can fine-tune auditing to match their specific security requirements. Auditing in Oracle has matured over decades, from its debut in Oracle Database 7 (1992) through the unified audit framework introduced in Oracle Database 12.1 (2012). Starting with Oracle AI Database 26ai, unified auditing is the only supported framework. By bringing previously separate audit

trails together into one normalized audit trail, teams can search once and get precise answers faster. Key advantages include:

- **Improved performance**. In typical configurations, audit-generation overhead is usually 3% or less for audit generation, even under heavy load of up to 200 audit records per second.
- **Stronger security**. The unified audit trail lives in the dedicated AUDSYS schema and is protected from DML operations, including update, delete, and truncate.
- **Greater flexibility**. Conditional auditing and an extensible audit trail make it easier to capture exactly what is needed and nothing more.

Note: Legacy auditing is desupported starting in Oracle AI Database 26ai. If an organization upgrades while still using legacy audit, the existing traditional audit settings will continue to be honored, and audit records will keep collecting in their respective trails to help bridge the gap to unified auditing. In Oracle AI Database 26ai, traditional audit settings cannot be created or updated and can only be deleted.

Transitioning from legacy to unified auditing

Legacy auditing (traditional auditing) is desupported in Oracle AI Database 26ai, which means new Oracle AI Database 26ai databases do not include the legacy audit facility. When you upgrade an existing database to 26ai, what happens next depends on where you are starting from.

If the source database is using unified auditing with no traditional audit settings, the upgraded database keeps running on unified auditing. If the source database still relies on legacy auditing, those traditional audit settings remain in effect after the upgrade, and audit records continue to land in the traditional audit trail until you turn them off with NOAUDIT. Also note that in 26ai you cannot create or update legacy audit policies; you can only delete the ones that already exist.

The practical takeaway is to plan the move to unified auditing either before the upgrade or immediately after, using the guidance below.

- Oracle Database provides predefined unified audit policies to help you get started. If you upgraded from release 11g, at a minimum enable these policies, which address common security and compliance requirements:
    - Security-relevant activity (ORA_SECURECONFIG), such as use of the CREATE USER system privilege
    - Logon failures (ORA_LOGON_FAILURES)

  All new databases created from release 12.2 and later automatically enable ORA_SECURECONFIG and ORA_LOGON_FAILURES.

If you have highly customized legacy audit settings, use the following transition options, listed in order of preference:

- Create custom unified audit policies that use the richer features of unified auditing to make your policies more conditional, selective, and focused. The unified audit facility supports fine-grained conditions and an extensible audit trail.
- If you are unfamiliar with unified audit syntax, use the syntax converter script in My Oracle Support note 2909718.1. It converts legacy audit configurations into syntactically correct unified audit policies and writes them to a script for your review. Oracle strongly recommends reviewing and enhancing the generated policies—add conditions or audit application context values—before you enable them.

After you convert legacy settings to unified audit policies, run NOAUDIT to disable traditional audit policies and remove traditional audit parameter settings. Update any purge jobs created with the DBMS_AUDIT_MGMT PL/SQL package to align with unified audit configuration.

Using audit data

Organizations often operate hundreds or even thousands of databases where both user and administrator activity must be audited and monitored to satisfy regulations and help detect security breaches. While monitoring database activity is essential, it can produce audit trails with substantial amounts of data. Because these audit trails are spread across many databases, analysis and reporting quickly become challenging. Once an organization moves beyond a handful of databases, audit data should be moved into a consolidated repository to support safekeeping, analysis, alerting, and report generation.

Database auditing is frequently strengthened with solutions that collect and store audit data from an entire fleet of databases. Oracle Data Safe and Oracle Database Security Central (Security Central) both provide this capability. In both cases, audit data is collected from the target databases and stored in a centralized audit repository. Both solutions include built-in information lifecycle management to manage retention and archiving of audit data, a wide set of audit reports with flexible filtering to meet requirements, and alerting features that highlight specific user actions or unusual behavior.

Both Data Safe and Security Central take a comprehensive approach, offering numerous benefits, including the following highlights:

- Centralize audit data in a dedicated repository to preserve data integrity and security
- Monitor database activity in near-real-time to capture audit records, analyze them against policies, and alert on violations
- Use built-in and customizable reports to meet security and compliance needs
- Aggregate and search audit records at scale to support forensic analysis
- Investigate breaches and suspicious activity with high-volume collection and efficient search across audit and network events
- Store and retrieve historical audit data
- Track user privilege changes to quickly identify risky escalation
- Demonstrate monitoring of critical systems and access to sensitive data with audit-trail analysis and auditor-ready reports to support regulatory compliance.

Audit data best practices

Collecting audit data is only the beginning. The real value comes from turning that data into action. As noted earlier, database activity is audited to do the following:

- Identify anomalous or suspicious activity
- Support regulatory compliance audits
- Assist in troubleshooting operations.

This is where analysis, reporting, and alerting become essential. The centralized repository created for audit data effectively serves as a data warehouse for activity across the entire database fleet. It can be used to do the following:

- Mine audit records for post-incident forensic analysis
- Generate alerts for unusual or policy-violating behavior
- Produce reports for stakeholders who need them.
- Accelerate troubleshooting by correlating events across systems

As AI agents increasingly generate queries autonomously, audit data becomes the primary mechanism for confirming that those agents remain within authorized boundaries. Agent-driven activity should be correlated against established

policies to verify that MCP servers and AI pipelines accessed only the data they were entitled to, and to flag any deviation for investigation.

Organizations that process sensitive data are often subject to regulations such as the General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), Payment Card Industry Data Security Standard (PCI-DSS), India's Digital Personal Data Protection Act (DPDPA), Health Insurance Portability and Accountability Act (HIPAA), IRS Publication 1075, Sarbanes-Oxley Act, and the UK Data Protection Act. Most regulations require visibility into who accessed sensitive data and a verifiable record of that access, often retained for several years.

Centralizing audit management

Database auditing is a foundational control for database security and regulatory compliance. In addition to defining audit policies, organizations also need to manage the audit data those policies produce. Audit data management typically includes the following:

- Controlling how much space audit data consumes in your databases
- Protecting audit records from tampering or deletion
- Retaining audit data to meet security goals and regulatory requirements.

The strongest approach is to collect audit data into a centralized repository. Centralization makes it practical to move audit data out of source databases, freeing space and easing day-to-day operational burden.

A dedicated repository also provides a more controlled environment to protect audit records from accidental changes and deliberate manipulation. It supports separation of duties by preventing users who administer or access the originating databases from altering the audit trail.

In addition, a centralized repository makes retention and lifecycle policies easier to apply consistently. Organizations can archive audit data to low-cost storage when it is no longer needed for reporting or analysis and delete older records when retention obligations no longer apply.

What to audit

Auditing every single action may sound like the safest option, but it often brings unintended costs: more storage, slower performance, and a flood of records that can drown out the signals that matter. In practice, most audit records are routine and add little security value, which can make malicious behavior harder to spot. A smarter approach is selective auditing that targets significant events. Strong audit policies capture the details that matter and deliberately skip routine actions from known sources. With less noise, teams can surface threats faster and respond with greater confidence.

In most cases, effective auditing keeps policies focused on the following types of database events while ignoring routine activity generated from known sources.

1. Security-relevant events
2. Privileged user activity
3. Sensitive data access

Oracle Database includes predefined audit policies for common requirements, making it easy to get started. Once the right policy is selected, the next step is to run the appropriate command to enable it. Here is an example of enabling one of the most commonly used policies:

```
AUDIT POLICY ORA_SECURECONFIG;
```

Figure 12-112: Enabling a predefined audit policy

The following sections review the three recommended event types and the policies that enable organizations to begin auditing quickly.

Auditing security-relevant events

Some database actions warrant closer monitoring because they can affect multiple schemas and may indicate misuse or attempts to hide activity. Audit the following security-relevant events:

- Failed login attempts. Audit all logins, with exceptions for allowlisted service accounts connecting from known locations and running approved programs.
- Schema changes. Audit `CREATE`, `ALTER`, and `DROP` operations on tables, views, PL/SQL program units, and similar DDL.
- Account and privilege changes. Audit `CREATE`, `ALTER`, and `DROP` operations for users, role and privilege grants, password changes, and related account management activity.
- Security policy changes. Audit changes to security policies, especially in schemas that contain sensitive data.
- Use of system privileges. Audit activity that relies on powerful system privileges (for example, `CREATE ANY TABLE` or `ALTER SYSTEM`).
- Large-scale data extraction. Audit bulk data movement using components such as Oracle Data Pump.
- Activity from dormant accounts. Audit connections and SQL activity from accounts that have been inactive for an extended period.

Three predefined unified audit policies help capture most of the activity above:

- ORA_SECURECONFIG
- ORA_LOGIN_LOGOUT (named ORA_LOGON_FAILURES in older database versions)
- ORA_ACCOUNT_MGMT

These policies cover the security-relevant events most organizations care about first. They can be enabled with a simple PL/SQL call (as shown in Figure 12-1) or through the audit management consoles in Oracle Data Safe or Oracle Database Security Central. The *Database Schema Changes* policy can also be provisioned from either the Data Safe or Security Central management console.

In addition, many other security-relevant events are always audited as part of the *Mandatory Audit*. These are activities that could materially affect database security or availability. For the complete list of auditable events covered by the mandatory audit, see the Oracle Database product documentation.

Privileged user activity auditing

Because privileged accounts often carry the highest risk, monitoring them is one of the fastest ways to spot unusual behavior and detect sensitive data leaks if an account is compromised or misused. From an auditing perspective, any database user granted system privileges such as SELECT ANY TABLE or ALTER USER should be treated as privileged, with activity monitored for potential abuse or misuse.

It is also important to note that top-level statements by administrative users (for example, SYSDBA, SYSKM) issued while the database is in the closed or mount state are included in the mandatory audit. Mandatory auditing is enabled by default and cannot be disabled.

Note: In this context, "top-level" refers to the statement the user actually enters. Auditing top-level statements captures only those statements, not the internal or recursive SQL generated as a result. For example, if top-level statements are audited and a DBA runs DBMS_STATS.GATHER_DATABASE_STATS, only the call to execute the PL/SQL package is audited. This approach avoids capturing the SQL executed inside named procedures and functions, which could otherwise generate thousands of audit records, and it significantly reduces routine administrator noise.

To capture top-level actions by administrative accounts such as SYS during normal database operations, enable additional audit policies. If Data Safe or Security Central is used to monitor database activity, provision the *Admin Activity Auditing* policy to audit activities by privileged administrators, including SYS.

Database connections can be made through the database listener (remote) or locally from an account logged into the database server. Auditing local sessions is critical because they can bypass network monitoring and may connect to the database without any authentication beyond the database server operating system account that initiates the session. Oracle recommends auditing all top-level operations that originate from a local session. Create an audit policy as given below to capture all top-level actions with appropriate audit conditions and consider enabling it for interactive users who have direct access to the database.

```
CREATE AUDIT POLICY local_db_access
ACTIONS ALL
WHEN '(SYS_CONTEXT (''USERENV'', ''IP_ADDRESS'') IS NULL)'
EVALUATE PER SESSION
ONLY TOPLEVEL;
```

Figure 12-2: Auditing local access to the database

When there is a need to monitor every action taken by privileged accounts that access sensitive data, the most effective approach is to target those users directly. With Data Safe or Security Central, the *User Activity Auditing* policy can be provisioned to audit all activity for a defined set of users. Another option is to create a custom unified audit policy that captures everything a user does. Figure 12-3 provides an example: the first statement creates a policy that audits all actions, and the second statement enables that policy for the user DBA_DEBRA.

```
CREATE AUDIT POLICY actions_all_pol
ACTIONS ALL
ONLY TOPLEVEL;


AUDIT POLICY actions_all_pol BY DBA_DEBRA;
```

Figure 12-3: Example syntax for auditing all activity by a user

It is important to keep in mind that auditing everything a user does can dramatically increase the volume of audit records, which can drive up storage costs and add performance overhead.

Sensitive data access auditing

Auditing access to, and updates of, sensitive data helps teams detect inappropriate activity sooner and helps discourage people from accessing or changing data when they do not have a clear business reason.

A practical approach is selective auditing of sensitive data access, which provides visibility into access and updates without generating unnecessary noise. The first step is to scan the database for sensitive data using Data Safe, Security Central, Database Security Assessment Tool (DBSAT), or Oracle Enterprise Manager. Chapter Three provides more detail on sensitive data discovery.

Once sensitive data has been found, a custom audit policy can be created to audit access outside normal application behavior. Figure 12-4 shows an example that audits all activity against the employees table in the HR schema (which

is presumed to contain sensitive data). The first statement creates the policy and identifies the object to monitor. The second statement enables the policy and applies it to all users.

```
CREATE AUDIT POLICY actions_on_hr_emp_pol
ACTIONS ALL
ON hr.employees;


AUDIT POLICY actions_on_hr_emp_pol;
```

Figure 12-4: Example syntax for auditing activity on the EMPLOYEES table containing PII

The policy above will likely capture routine access from the HR application, which can create a large volume of low-value audit records and introduce additional performance and storage overhead. A better option is to audit only access to the table that happens outside the approved application. The updated example in Figure 12-5 filters most routine activity and focuses the audit trail on meaningful events.

```
CREATE AUDIT POLICY actions_on_hr_emp_pol
ACTIONS ALL
ON hr.employees
WHEN 'SYS_CONTEXT (''USERENV'', ''HOST'') NOT IN (''APPSERVER1'',''APPSERVER2'')'
EVALUATE PER SESSION;


AUDIT POLICY actions_on_hr_emp_pol;
```

Figure 12-512: Example syntax for auditing activity from outside of an application

In the modified policy, a WHEN condition is added to treat access through the application server as the trusted path. The policy checks the session context value HOST to confirm the connection came from the application server. To keep overhead low, the condition is evaluated once per session using EVALUATE PER SESSION, and the result is reused for the rest of that session.

As a result, activity on the table that does not come from the application server generates audit records, while routine application server activity does not. The WHEN condition can use complex AND/OR logic, but it is limited to 4,000 bytes. If more logic is required, a custom PL/SQL function can be used. For more complex conditions, the EVALUATE PER SESSION clause can greatly reduce overhead.

A practical strategy is to focus audit policies on privileged user activity, security-relevant events, and sensitive data access. This keeps auditing centered on what matters, cuts down unnecessary audit records, and still supports audit goals.

Because audit policies are flexible by design, teams can target exactly what is needed while filtering out routine activity that drags down performance, increases storage costs, and makes unusual or malicious behavior harder to spot.

Effective audit policies

Here are general guidelines for designing effective audit policies:

1. **Do not duplicate mandatory audits**—Creating policies that capture audit information already included in mandatory auditing can have a slight but measurable performance impact because audit policies are evaluated as commands run. Refer to the Oracle Database Security Guide for the complete list of mandatory auditable events for your Oracle Database release. For example, avoid duplicating mandatory audit coverage in custom policies.

2. **Use predefined audit policies**—Oracle Database includes several predefined best-practice unified audit policies that cover common security-relevant audit settings. Enable these policies to start collecting audit data. Available policies vary by release, so check the Oracle Database Security Guide for your release. For example, the guide describes the predefined audit policies included with Oracle AI Database 26ai. If you use Data Safe or Security Central, you can enable many recommended audit configurations and predefined Oracle Database audit policies with one click or a REST API call.
3. **Create custom audit policies for context-dependent scenarios**—Some configurations are unique to your environment—for example, access to sensitive data—so create custom unified audit policies in Oracle Database for those cases.

Auditing in the age of AI

As agentic AI systems become more autonomous and start executing complex tasks at machine speed, auditing moves from a nice-to-have to a practical necessity. By watching how agents invoke tools, share context, and run tasks through MCP pipelines that interact with Oracle Database, organizations can illuminate the data access patterns that sit behind autonomous actions.

Unified auditing helps deliver end-to-end visibility into how agentic AI accesses the database through MCP pipelines. When an agent sends MCP orchestrated SQL into Oracle Database, unified auditing captures the executed statements along with context, user identity, and source details. That makes it easier to detect unusual access patterns, unauthorized privilege usage, or abnormal data extraction that automation may trigger.

Oracle SQLcl MCP Server also makes tracing easier by identifying itself in the database session using the PROGRAM, MODULE, and ACTION properties in V$SESSION. MCP Server interactions can then be logged to UNIFIED_AUDIT_TRAIL using the statement shown in Figure 12-6.

```
AUDIT CONTEXT NAMESPACE userenv ATTRIBUTES module, action;
```

Figure 12-612: Example syntax for auditing attributes from USERENV

The sample unified audit record below captures an interaction between Oracle SQLcl MCP Server and a GitHub Copilot agent, in which the agent queries data in Oracle Database using SQL generated by Claude.

	CLIENT_PROGRAM_NAME	SQL_TEXT			APPLICATION_CONTEXTS
1	SQLcl-MCP	SELECT /* LLM in use is claude-3.5-sonnet */	EMPLOYEE_ID,	…	(USERENV,ACTION=run-sql); (USERENV,MODULE=claude-3.5-sonnet)

Figure 12-712: Example syntax for auditing Agentic AI interactions

Managing and using audit data with Oracle Data Safe

Organizations subject to regulations must often demonstrate consistent monitoring and timely analysis of audit data. An audit data warehouse supports these obligations by automating retention policies and producing auditor-ready reports, helping organizations demonstrate consistent monitoring and timely analysis.

Two proven options are available to centralize, analyze, and act on audit data at scale:

1. **Oracle Data Safe.** A database security cloud service running in Oracle Cloud Infrastructure (OCI). Data Safe helps manage day-to-day security and compliance for Oracle AI Databases, both on premises and in the cloud. It includes policy management and retention, centralized audit collection, a broad library of reports with flexible filters, and alerting for specific activities or unusual behavior, helping teams identify issues faster and streamline reporting.

2. **Oracle Database Security Central.** Deploy on premises or in the cloud to unify database auditing with network-level SQL traffic monitoring. Security Central combines native audit data collection with network-based capture for commonly used databases. It provides a highly scalable, enterprise-grade data warehouse, collection agents, Database Firewall and SQL Firewall, robust reporting and analytics, and a flexible alert framework. Recommended auditing policies can be applied with a single click to accelerate time to value.

The rest of this chapter explains how to manage audit data volume and build a centralized repository that delivers the analysis, reporting, and alerting needed for security and compliance.

With Oracle Data Safe, audit records are centralized from Oracle AI Databases, including SQL Firewall violations (see Chapter Eight). After audit data is collected, it can be safely deleted from the source databases to reclaim space and reduce overhead. In Figure 12-8 below, a fleetwide retention policy is configured in Data Safe. Audit records can be retained for up to seven years, up to one year online and up to six years in archive storage, helping meet compliance requirements without overloading databases.

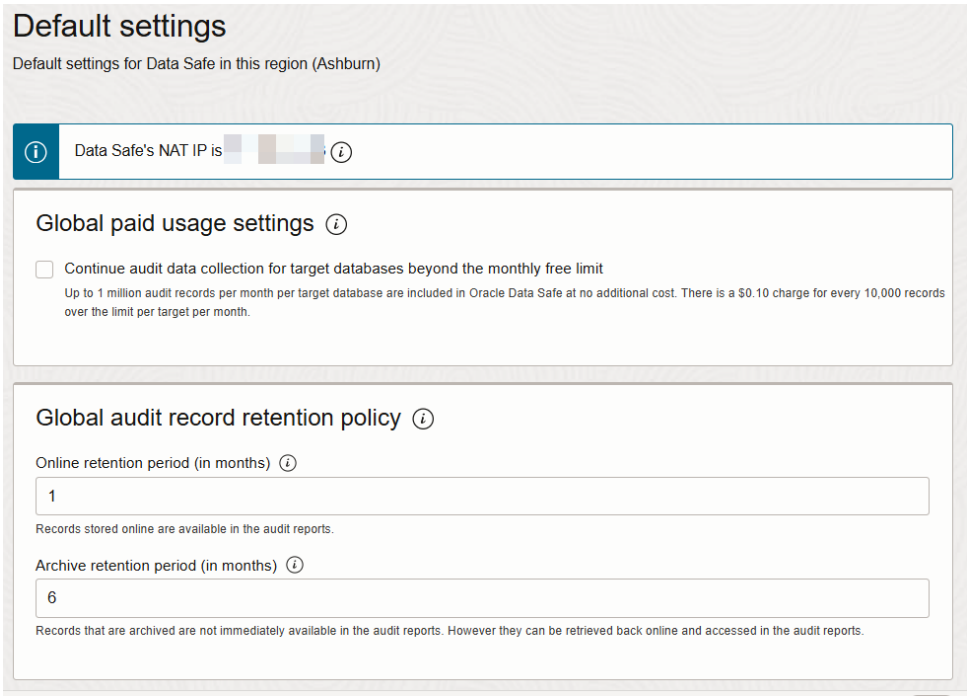


Figure 12-8: Data Safe audit data global retention settings

As shown in the following figure, the fleet setting can be overridden for individual databases that require different retention periods.

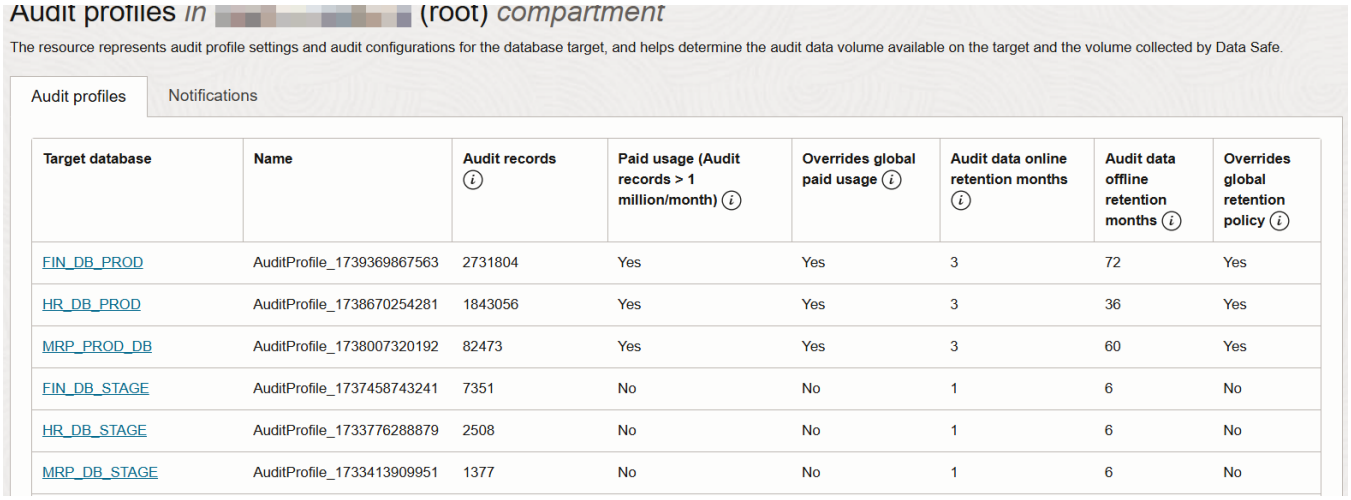


Figure 12-9: Data Safe individual database retention settings

Data Safe provides interactive reports that help track specific users, objects, or actions. It also supports forensic investigations and delivers summary reports for compliance and activity reporting. Reports can be scheduled, downloaded as PDFs, and run on recurring schedules to document periodic compliance reviews without manual effort. To stay ahead of risk, teams can choose predefined alert policies that flag unusual activity that may indicate compromise or create custom alert policies tailored to the environment.

Data Safe supports Oracle AI Database across deployments in Oracle Cloud Infrastructure, Cloud@Customer, on-premises, and other clouds such as AWS or Azure.

After Data Safe provisions and enables audit policies for a target database, it collects audit records for the activities being audited. To keep source databases from retaining too much audit data, the optional auto-purge feature (disabled by default) can be enabled to delete audit data from the target after collection.

Audit insights

Audit Insights provide a fast way to surface patterns and outliers across the fleet, helping teams focus on investigations, tune policies, and report with confidence. With Audit Insights, organizations can do the following:

- Spot spikes in Data Definition Language (DDL) and Data Manipulation Language (DML) activity
- Track trends in failed logins and user or entitlement changes
- See which databases, schemas, and objects contribute most to audit volume
- Find the audit policies that generate the most records
- Identify client connections and database users that generate the most audit volume
- Verify that audit policies capture the intended activity on target schemas and objects
- Filter results by time period and target scope to keep analysis focused on investigations.

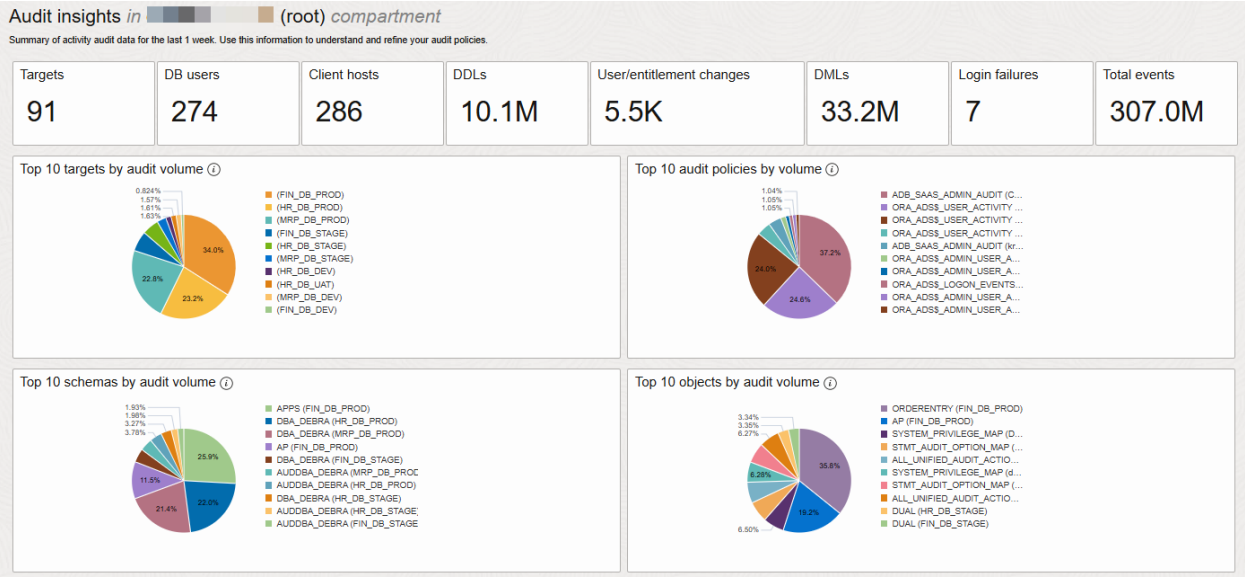


Figure 12-10: Data Safe audit insights

Audit Insights metrics and charts can be used to examine audit record volume and quickly spot outliers. By analyzing the top contributors by audit volume, including databases, schemas, objects, users, client connections, and policies, teams can pinpoint where auditing should be tightened or streamlined. Adjusting noisy or incomplete policies reduces overhead, keeps attention on high value activity, and strengthens the overall security of target databases.

Activity auditing dashboard and reports

The activity auditing dashboard presents charts and tables that summarize audit events from the past week across all target databases. Filters can be used to adjust the time range, focus on all databases within a specific compartment, or drill into a single target database for deeper analysis.

Audit records should be centralized in the Data Safe repository to support confident analysis and reporting. Once data is in the repository, interactive reports are available for user activity tracking and forensic investigation, along with summary reports for routine compliance and operational reporting.

Data Safe includes a library of predefined audit reports that surface detailed database activity, including the following:

- Privileged user activity
- Login activity
- Schema changes
- Entitlement changes
- Access to sensitive data.

These reports help teams quickly spot issues, investigate incidents, and document oversight. See Table 12-1 for examples of predefined audit reports available in Data Safe.

All activity	**All audit activities**
Admin activity	Database activities of admin users (as identified in Data Safe's user assessment feature)
User/entitlement changes	User creation and deletion, privilege and role changes

Audit policy changes	All changes in audit policies
Login activity	Database login attempts
Data access	Database query operations
Data modification	Data modification activities (DMLs)
Database schema changes	Database schema changes (DDLs)
Common user activity	Database activities of common users in pluggable databases
Database errors	Errors reported in the database for audited activities
Data extraction	Data Pump and RMAN activities in the database
Sensitive data activity	Database activity on sensitive objects (as identified in Data Safe's data discovery feature)

Table 12-1: Data Safe audit reports

Filters can be set and report columns can be adjusted to meet specific needs. To reuse a report with these customizations, the modified report can be saved as a custom audit report. Predefined or custom reports can also be scheduled to run at specific times, with results available for download as PDF or XLS files.

Audit retention

Audit retention plays a central role in meeting organizational policies and regulatory requirements, which commonly require audit records to be preserved for a defined period, often several years. This ensures compliance while maintaining a dependable history for future investigations, audits, or reviews.

Data Safe uses a two-tier repository model: online storage for timely reporting and analysis, and offline archive storage for long-term retention. Retention settings can be applied globally across all databases or adjusted for specific databases that require a different policy. When set, individual database retention settings override global settings.

Online retention ranges from one month to 12 months. Offline retention ranges from zero months to 72 months. Together, these settings allow audit records to be retained for up to 84 months (seven years). For retention beyond seven years, a request can be submitted through Oracle Support.

Data Safe automatically transitions audit data from online to offline storage when the online retention window ends and permanently deletes the data once the offline retention period expires.

When older records are needed, archives can be retrieved from offline storage into the Data Safe online repository for analysis and reporting. Retrieved archives remain available for up to one month before returning to the archive automatically, and they can be released back to the archive manually at any time.

Alerts

Alerts can be used on target databases to monitor high-priority activities and notify the right stakeholders when events occur. Data Safe generates an alert when a specified audit event occurs on a target database, according to the alert policies that have been enabled. Teams can enable multiple preconfigured alert policies, or create custom policies tailored to their environment. See Table 12-2 for the list of preconfigured alert policies.

Audit policy	Severity level
Failed logins by admin users	Critical
Profile changes	Critical
Audit policy changes	High
Database parameter changes	High
Database schema changes	Medium
User creation/modification	Medium
User entitlement changes	Medium

Table 12-2: Alert policies in Oracle Data Safe

Create custom alerts to scan incoming audit data and apply filters that detect specific conditions. For example, as shown in Figure 12-11, an alert can be triggered when an audit record involves the EMPLOYEES table and the actor is not the application account.

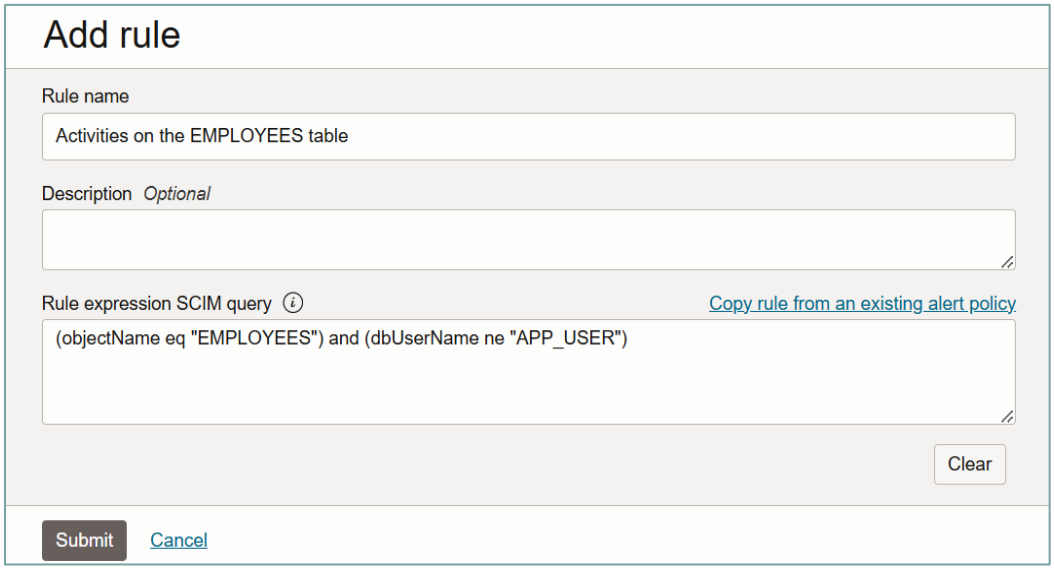


Figure 12-11: Data Safe example of a custom alert policy

Use System for Cross-domain Identity Management (SCIM) style filters to define alert conditions and report filters with a consistent, flexible query syntax.

Data Safe also provides a built-in alert dashboard for viewing and managing alerts. For notifications, Data Safe alert events can be connected to Oracle Cloud Infrastructure (OCI) Events and Notifications to route messages to selected channels (for example, email).

Auditing with Oracle Database Security Central

Security Central is a scalable, flexible solution for database auditing and network activity monitoring. It centralizes audit data and surfaces risk sooner. It consolidates audit records from databases, operating systems, directories, file systems, and applications into a single repository for analysis, alerting, and reporting. Security Central is deployed as a full stack software appliance on dedicated hardware or on virtual machines.

Security Central supports the most common database types, including Oracle AI Database, MySQL, Microsoft SQL Server, SAP Sybase, IBM Db2 LUW, PostgreSQL, and MongoDB. It can also ingest almost any standard format audit source using audit collection plugins that read from database tables and CSV, XML, or JSON files. For targets that are not accessible through the quick collector framework, the included software development kit (SDK) can be used to build custom collectors.

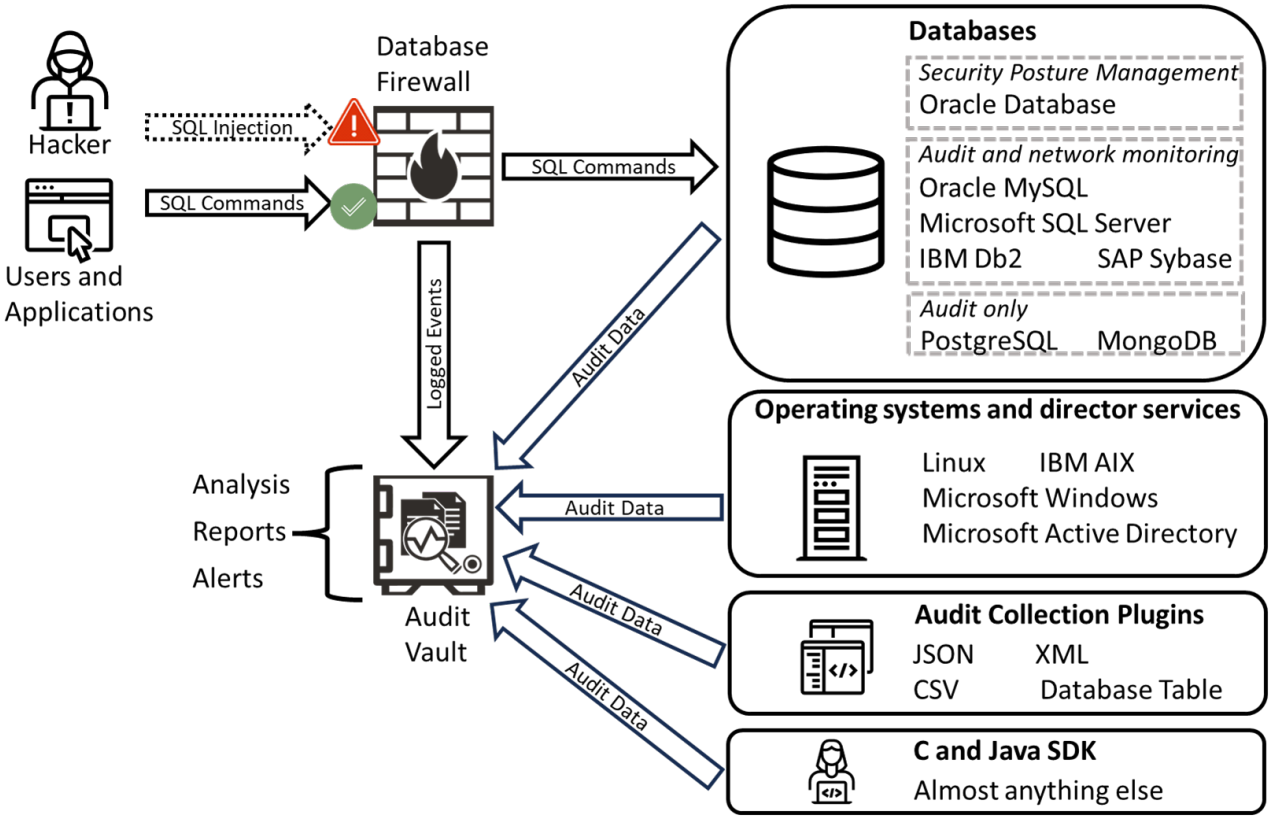


Figure 12-12: Functional architecture showing Security Central data flows

External monitoring

Native auditing captures only the activity that is explicitly defined in audit policy. That precision keeps noise down, but it can also miss first-time clients or unexpected SQL. External monitoring addresses that blind spot. AI agents and MCP servers often introduce novel SQL, either by generating queries that were not anticipated in existing policies or by reaching beyond their intended table scope. Database Firewall captures this traffic even when no audit policy exists for it, creating an added layer of visibility for agent-driven activity that does not match established patterns.

Database Firewall (see Chapter Eight) can log traffic that falls outside the policy allowlist, including new clients, unseen SQL, and other outliers. Although these records may not contain full session context, they are valuable for anomaly detection and efficient triage.

Activity data collection

The Audit Vault Server collects audit data from databases, operating systems, and application tables or files, and it ingests events from Database Firewall and policy violations from SQL Firewall. It then maps all collected data to a normalized schema and stores it in Security Central's repository, a specially configured Oracle AI Database. By normalizing the data, Security Central enables a unified reporting view across heterogeneous sources, from databases and operating systems to applications and directories.

For Oracle and Microsoft SQL Server, Security Central can capture before and after values, preserving a clear history of who changed data and the previous and updated values, which supports data governance. For Oracle AI Database, Security Central also evaluates security configuration, including drift detection, tracks user entitlements, and monitors stored procedure changes, as discussed in Chapter Three. When teams are unsure what to audit, Security Central offers recommended audit policies for Oracle AI Database that can be provisioned in a few clicks.

The Audit Vault Agents collect audit data from target audit trails, including databases and operating systems, and forward it to Audit Vault Server. Audit Vault Agents poll targets on a schedule, collecting only records added since the last run and sending only the deltas. One agent can collect from multiple target types. Security Central also supports agentless collection for Oracle AI Database and Microsoft SQL Server, pulling audit data directly from targets without installing an agent on target host machines.

Audit reports

Security Central offers dozens of out-of-the-box reports that illuminate database activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before and after data value changes, and login failures. Predefined compliance reports are also available for GDPR, PCI-DSS, SOX, HIPAA, Gramm-Leach-Bliley Act (GLBA), DPA, and IRS Publication 1075.

Security Central reports provide detailed insight into database activity events collected from both audit data and the Database Firewall (see Chapter Eight for more details on the Database Firewall). Coverage includes all audit and network activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before and after data value changes, sensitive data access, and login failures.

Table 12-3 summarizes the different types of reports in Security Central.

Report type	Description
Activity	These reports track database access activity, including audited SQL statements, application access, and user logins. Typical activities include: • All activity • Data modifications • Before and after values of modified data • DDL activity • Failed logins
Alerts	Alert reports display the raised alerts and their status.
Stored Procedure Audit	This report tracks stored procedure changes, including create, replace, and drop.
Database Firewall	These reports detail SQL traffic monitored by Database Firewall, including statements that generated warnings or were blocked by the database firewall policy.
User Entitlements	These reports show user privilege information for Oracle Database targets. Security Central supports establishing a baseline and reporting drift from it, making it easier to monitor and investigate changes.
User Correlation	For Oracle Database targets on Linux, these reports correlate database events with the original Linux OS user. This is especially useful when users log in with named accounts and then switch to special purpose users, such as the database owner, usually ORACLE, using *su* or *sudo*.
Database Vault Activity	The Database Vault Activity report shows Database Vault events, which capture policy or rule violations and unauthorized access attempts.
Anomalous Activity	These reports pertain to new or previously dormant users accessing the system, or users accessing the system from IP addresses not seen before.
Assessment Reports	These reports bring together security assessment data from Oracle AI Databases and provide recommendations designed to strengthen the security of the Oracle AI Database system. They also include drift reporting against an established security baseline.
SQL Firewall Violation Reports	The report details SQL Firewall violations, which indicate abnormal access patterns that are not in the allow-lists defined in SQL Firewall policy.

Table 12-3: Built-in reports in Security Central

Auditors can interact with Security Central reports in a web interface and export results as PDF or XLS files. Security Central also supports scheduling reports to run automatically and distributing them by email. Because the Security Central repository is built on an Oracle AI Database and exposes an open, documented schema, organizations can query it using their preferred reporting tools and integrate it with platforms such as Splunk or their SIEM.

Audit insights

The Audit Insights dashboard offers a fast, high-level view of activity captured from auditing and network-based events across one or many databases. It is designed to start with a high-level overview and then support deeper analysis through clickable charts and tables. Events can be explored by context, such as database target, user, privileged user, event type, and more, to sharpen investigations and streamline reporting as shown below.

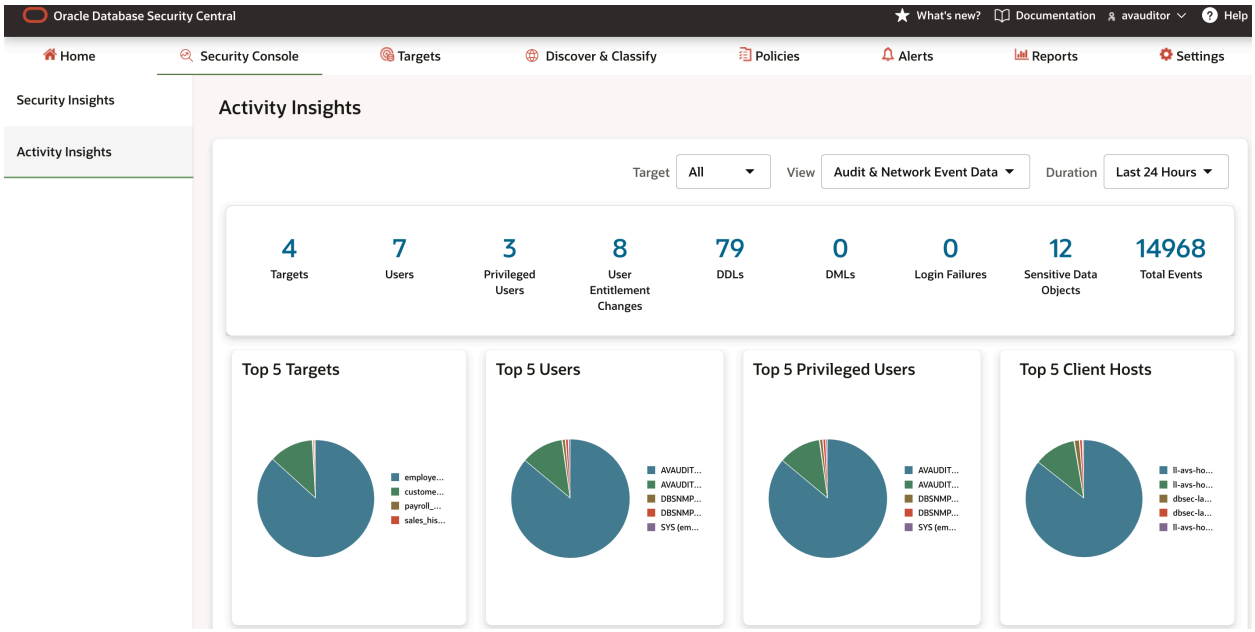


Figure 12-13: Security Central Audit Insights

Before-and-after data value changes

A key strength of Security Central is its ability to follow data changes end to end, making it much easier to show who changed what and when. Each time an update occurs, Security Central records both the original value (before) and the new value (after), along with the actor and timestamp.

That level of detail offers a significant advantage for organizations aiming to meet strict data governance expectations, especially in heavily regulated sectors like healthcare and financial services. In fact, some regulators, including India's Ministry of Corporate Affairs, require before and after tracking for financial data. Auditors routinely lean on these records to trace the lifecycle of specific attribute changes and to confirm that the right controls are in place and working properly.

Before and after value tracking is available for Oracle and Microsoft SQL Server databases.

Security Central's special support for Oracle AI Databases

Audit collection works across many sources, and network event collection covers most major databases. Security Central also includes capabilities designed specifically for Oracle AI Database as part of its database security posture management (DSPM) features. The SQL Firewall feature in Security Central lets you administer and monitor SQL Firewall across your fleet of Oracle AI Database 26ai targets.

Provisioning audit policies

Audit scope should align with an organization's policies and risk profile. In practice, this typically includes privileged activity, security relevant events such as user creation and privilege or role grants, changes to database structures, and access to sensitive data, especially access that occurs outside the application.

Audit policy design requires balance: capture enough detail to satisfy regulations and support investigations, while keeping policies targeted to limit noise, performance impact, and storage costs. Security Central supports this by offering a baseline set of recommended policies for Oracle AI Database that can be enabled with a single click. It also provides preconfigured compliance policies aligned to standards such as CIS and STIG.

Figure 12-14 provides a glimpse of audit policy provisioning workflow that offers **ready-to-use policies which can be provisioned with a single click.**

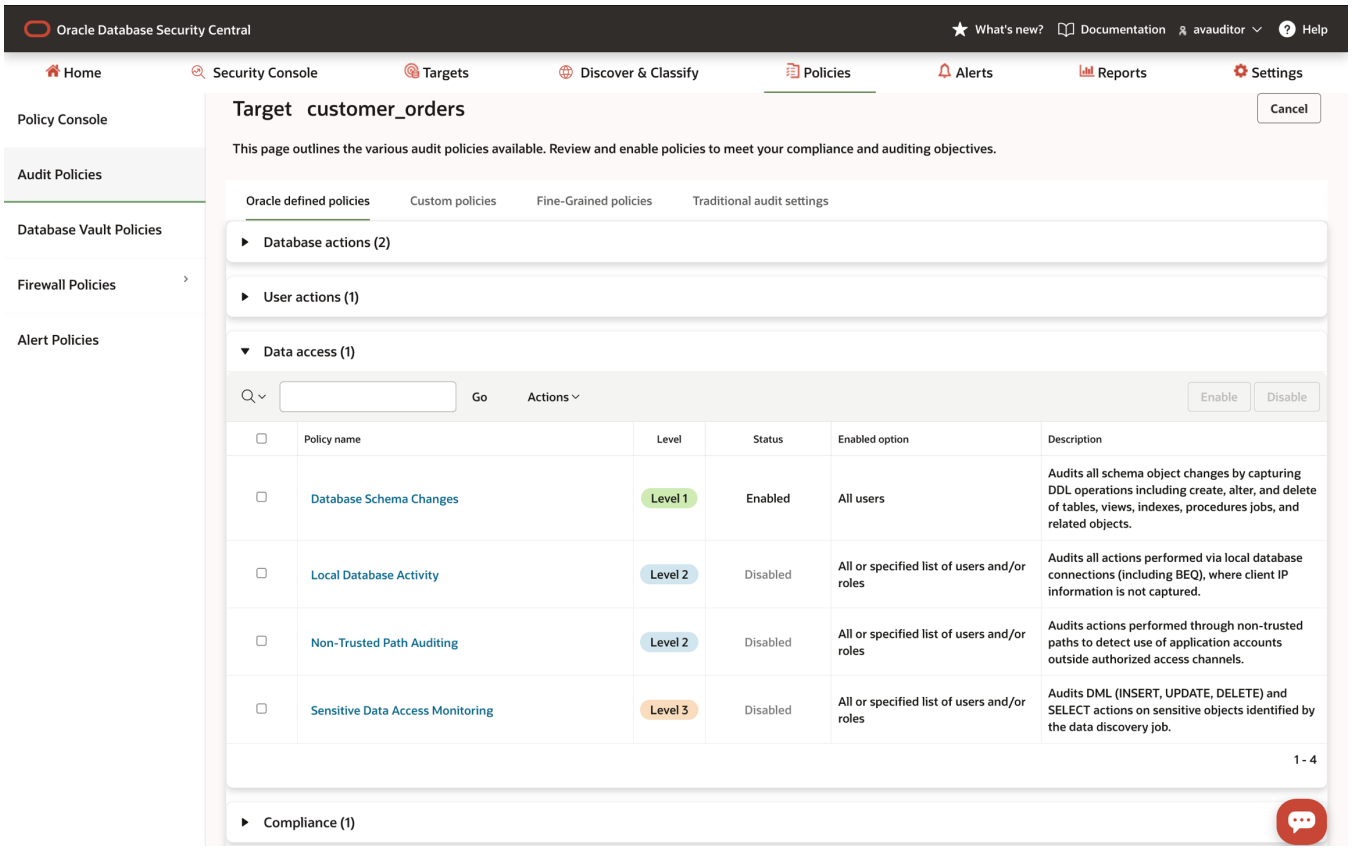


Figure 12-14: Security Central Audit Policy provisioning

Database security configuration, user, and sensitive data risk assessment

As discussed in Chapter Three, Security Central assesses database security configuration risks, and detects configuration drift. Fleet level reporting helps teams quickly spot configuration issues and decide which databases should be addressed first.

Chapter Three also covered how Security Central assesses user risks and tracks drift in privilege and role assignments. Security Central reports list each user's privileges and roles and can compare results to a baseline or an earlier snapshot to highlight what changed. This supports a certify once, then monitor approach that reduces manual effort and lowers the risk of human error in periodic reviews.

Chapter Three also covered how Security Central discovers the sensitive-data landscape across the fleet. Security Central classifies data based upon table metadata and actual data. In addition to supporting 181 sensitive types across 20 categories, Security Central supports custom sensitive-data types, and applies consistent classification across the fleet.

Drift detection for stored procedures

Stored procedures often encapsulate sensitive business logic and data access, which makes them an attractive target for attackers seeking to introduce backdoors. Security Central periodically checks Oracle AI Databases for new or modified stored procedures and generates reports for review, helping teams spot potential tampering quickly when compromise is suspected.

Monitoring stored procedure modifications helps detect a common post-compromise attacker technique: backdoor injection into legitimate stored procedures.

After gaining initial access, sophisticated attackers modify existing stored procedures to create persistent backdoors. Rather than creating new, obviously suspicious accounts or procedures, they inject malicious code into established, trusted procedures that execute with elevated privileges. These modifications typically occur outside normal patching windows, making them detectable through change monitoring.

Stored procedure auditing enables organizations to:

- **Determine exactly when procedures changed**. Timestamp tracking identifies modification timing
- **View code differentials**. Side-by-side comparison shows what changed between versions
- **Differentiate expected updates from unexpected drift**. Changes during maintenance windows are expected; changes at 2 a.m. on weekends trigger investigation
- **Focus on high-risk procedures**. Prioritize monitoring procedures that access sensitive data or execute with elevated privileges

Figure 12-15 shows the stored procedure auditing configuration interface where administrators schedule monitoring frequency and scope.

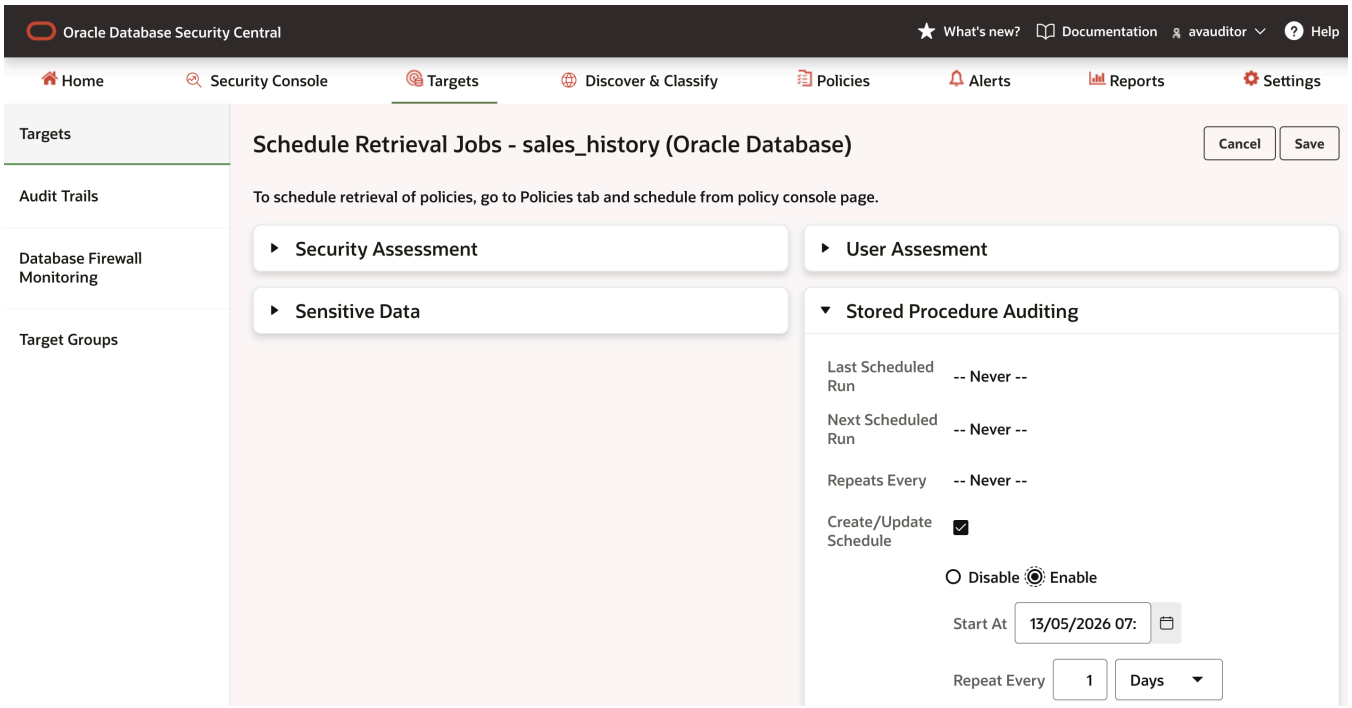


Figure 12-15: Example Security Central configuring stored procedure auditing

When Security Central detects stored procedure changes outside maintenance windows, it generates alerts with code differentials, enabling rapid investigation. This targeted change tracking helps organizations respond before hidden backdoors escalate into data breaches.

Sensitive data discovery

Security Central helps organizations find and classify sensitive data so they can see where risk is concentrated and quantify how much sensitive data they have. As discussed in Chapter Four, it can discover sensitive columns, label them by data type (for example, personal, financial, or health), and report volumes by database, schema, table, and column.

SQL Firewall

The SQL Firewall feature in Security Central lets you administer and monitor SQL Firewall across your fleet of Oracle AI Database targets. SQL Firewall is a new security feature built into the Oracle AI Database 26ai kernel that mitigates risks from SQL injection attacks, anomalous access, and credential theft or abuse. SQL Firewall inspects all incoming database connections and SQL statements, including the ones from PL/SQL units, whether local or over the network, encrypted or clear text. It only allows explicitly authorized SQL and can log or block SQL statements and connections that do not fall within the SQL Firewall allowlists.

SQL Firewall uses allowlists of authorized SQL statements and trusted database connection paths to determine which SQL statements and connection paths are authorized and which ones should be either logged or blocked. SQL Firewall allowlist policies work at a database account level. You create a SQL Firewall policy for a database account by learning or collecting the expected application SQL workload from expected database connections. Subsequently, the firewall detects and prevents unauthorized SQL and potential SQL injection attacks.

Figure 12-15 shows the SQL Firewall policy created for an application service account EMPLOYEESEARCH_PROD, and enforced in blocking mode.

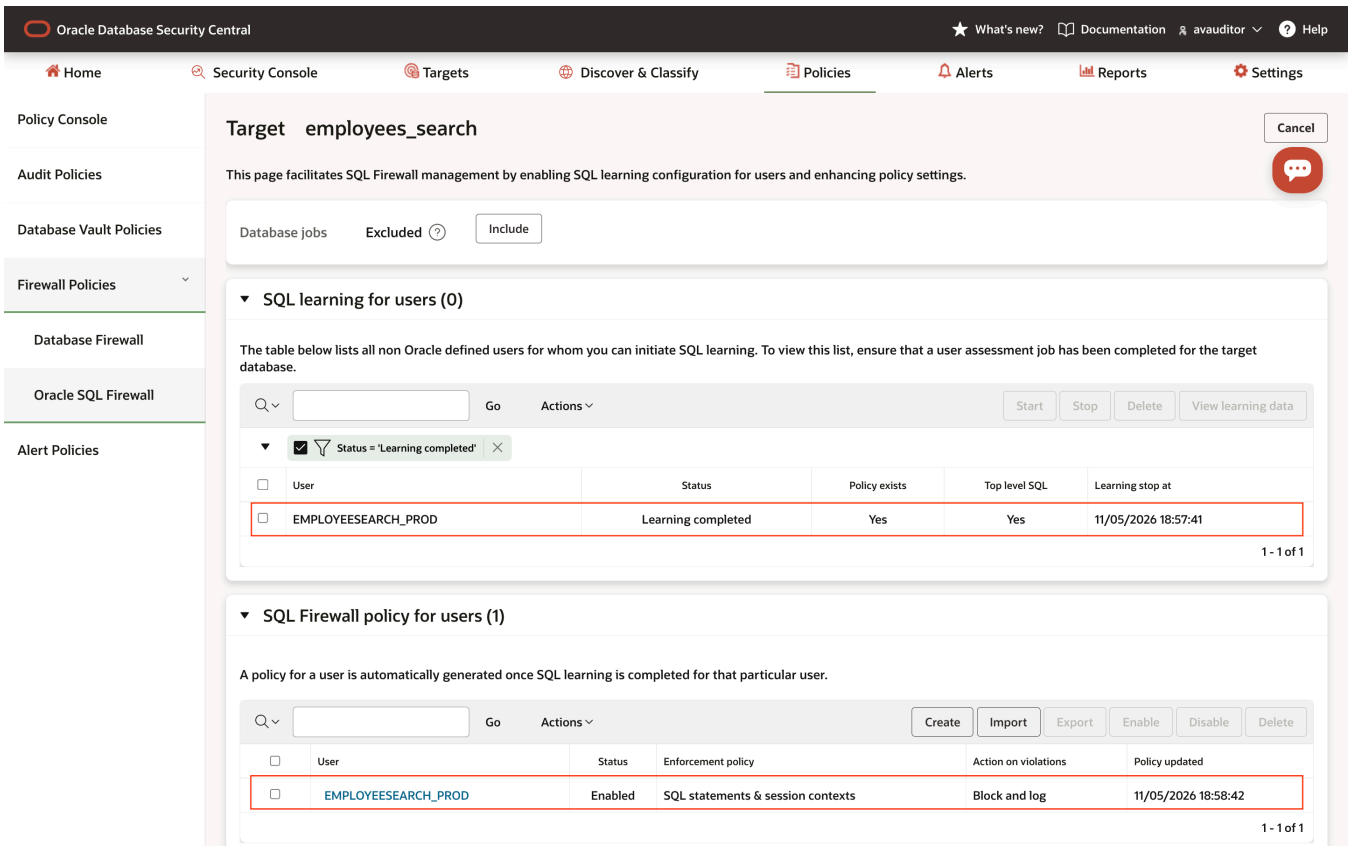


Figure 12-16: Security Central SQL Firewall Policy provisioning

SQL traffic that does not match the allow-list patterns enforced in the SQL Firewall policy will trigger SQL Firewall violations that can be monitored from Security Central as shown in Figure 12-16, and is also available as a report.

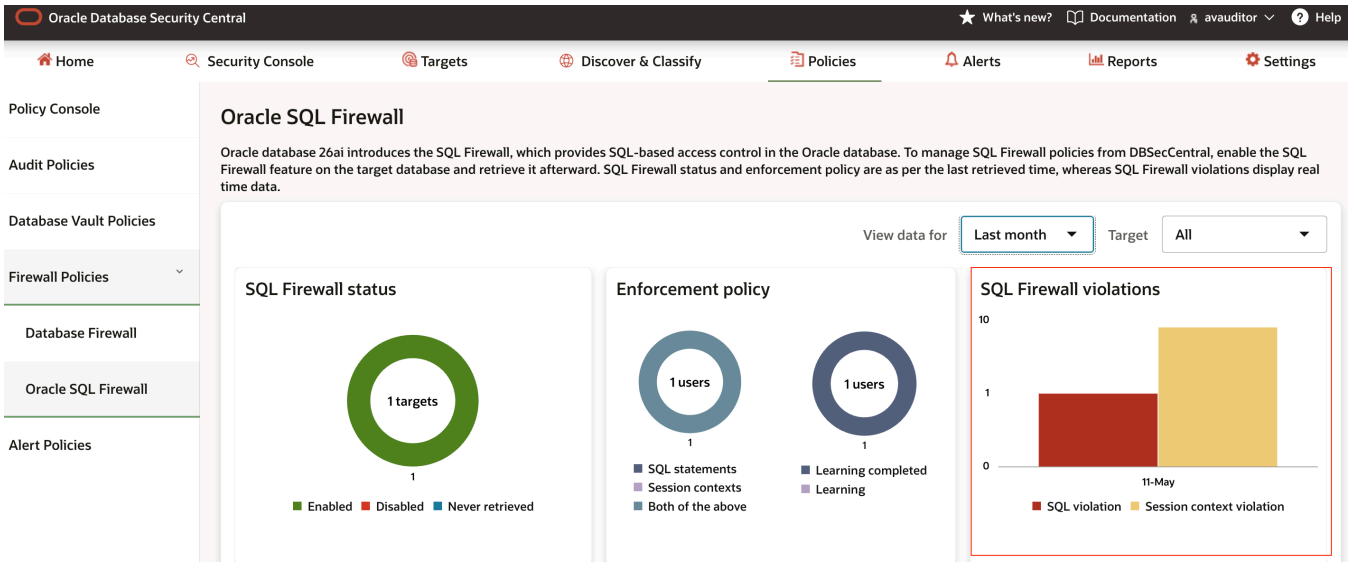


Figure 12-17: Database Security Central SQL Firewall Violations

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Get started with Oracle Data Safe fundamentals → Lab 6: Audit database activity
- Integrate Oracle Data Safe with Applications and Services → Labs 3, 4 and 5
- Oracle Database Security Central

Summary

Database auditing is a practical control that only delivers value when teams actively review the data and turn findings into action. It helps teams replace assumptions with evidence. Audit records become truly authoritative only when someone examines them, mines them for meaning, and uses them to drive decisions.

Auditing is worth doing because it helps organizations catch suspicious behavior earlier, build a defensible timeline during investigations, discourage risky actions, support non-repudiation, demonstrate compliance, keep a close watch on privileged accounts, troubleshoot issues such as repeated failed logins, and reconstruct prior states after an incident. A newer factor makes this even more important: AI agents and MCP-driven pipelines can generate floods of unexpected SQL, often through service accounts. If that activity is not captured and clearly separated, accountability can quickly blur.

On the technical side, standardize on Oracle Database unified auditing, which is the only supported framework starting with Oracle AI Database 26ai. Legacy auditing is desupported; in upgraded 26ai environments, traditional settings are limited to deletion. To move quickly, start with predefined unified audit policies such as ORA_SECURECONFIG and ORA_LOGON_FAILURES. If you have legacy customizations, convert those settings into custom unified audit policies, and then review and enhance the converted output before enabling it. Keep the design grounded: do not duplicate mandatory audits, prioritize event types that genuinely matter, and use conditions to prevent normal application noise from flooding the audit trail. For example, audit activity on hr.employees while filtering approved application hosts, and evaluate conditions once per session to keep overhead down.

Operationally, plan for centralized audit management. Scattered trails across a fleet quickly become difficult to report on, retain, and investigate. Oracle Data Safe can centralize audit records, support retention with online and offline tiers for up to seven years, optionally auto-purge on targets after collection, and provide Audit Insights alongside predefined reports and configurable alerts. Those alerts can use SCIM-style filters for custom conditions, and notifications can be routed through OCI Events and Notifications. For a broader approach, Security Central centralizes both audit and network activity monitoring via a software appliance deployment model. It maps diverse sources into a normalized repository schema, provides dozens of out-of-the-box and compliance reports, adds Database Firewall and SQL Firewall for deeper visibility. The firewall captures traffic that can slip past native audit policy scope, including first-time clients and the unexpected SQL that may accompany agent-driven activity.

With the audit foundation in place, the narrative naturally turns from visibility to restraint. The next chapter picks up where auditing leaves off by confronting how agentic workflows, vibe coding shortcuts, and shadow AI can reshape risk, especially when automation starts acting at machine speed with valid credentials. The proper approach is toward a layered, database-centric strategy that tightens the “who” and the “how” through least-privilege and runtime guardrails, keeps data unreadable by default through encryption and external key management, and constrains what AI can see through controls such as Database Vault, VPD, OLS, and RAS, backed by monitoring and blocking capabilities like Database Firewall and SQL Firewall. For teams that want to move fast while maintaining governance, the next chapter is the map from bright ideas to hardened practice, turning AI ambition into governed execution with Oracle Database security controls as the control foundation.

Chapter 13

Securing Agentic AI

Security risks in the age of AI

Artificial intelligence is transforming industries, driving innovation and reshaping day-to-day business operations. What differentiates this moment is not only AI's ability to generate text or code, but its growing ability to act by planning work, executing tasks, and interacting with enterprise systems. As agentic AI systems expand across enterprises, they bring major productivity gains while extending the security perimeter in ways that many organizations have yet to map, measure, and manage.

AI-generated coding accelerates delivery, but speed can create security risk if it is not carefully managed. Quickly produced code can introduce vulnerabilities that are hard to detect, reproduce, or trace, especially when developers accept suggestions without full validation. In parallel, AI assistants embedded in development pipelines, ticketing systems, and deployment tooling can introduce new opportunities for abuse, including prompt injection, insecure tool use, accidental exposure of secrets, and the quiet spread of flawed patterns across multiple services.

Risk increases when AI goes beyond generating code and starts working directly with enterprise data and systems. AI that can access data, process it, and make changes is powerful, but it can be risky without proper security. Autonomy without guardrails can lead to data overreach, unintended disclosure, or destructive actions, particularly when permissions are overly broad for convenience, access is not well segmented, or auditability and policy enforcement fall behind the technology's capabilities. A sustainable approach pairs autonomy with strong data safeguards so innovation remains under control.

Enterprise security strategy therefore needs to evolve from protecting static applications to governing dynamic, AI mediated actions. Organizations benefit from security built for AI-era workflows, including least-privilege data access, strong identity controls, reliable auditing, and governance that can keep pace with automation and orchestration.

To manage these emerging risks without slowing delivery, organizations need controls that sit close to the data and work across both human and AI-driven access paths. Organizations can tap into AI's momentum while keeping the business protected with proven, enterprise-grade database security. With Oracle AI Database 26ai, orchestration layers like MCP servers, and Private Agent Factory, a no-code platform that lets teams build and deploy AI agents entirely within their own environment, AI can move fast while governance stays firmly in place.

Modern risks: vibe coding and shadow AI

AI-accelerated development has made it easier than ever to build systems without fully understanding what was built or what risks those systems carry. This is the backdrop for "vibe coding," a growing practice in which developers lean on AI assistants to generate entire functions, classes, or even application architectures from natural language prompts, rather than from deep technical understanding.

While this democratization of coding can unlock remarkable productivity gains, it also creates a serious blind spot: code that appears to work flawlessly while its underlying mechanics remain unclear to the humans responsible for operating it.

The security implications are serious. Unvetted AI-generated code can introduce injection vulnerabilities, expose API keys through improper error handling, implement flawed authentication logic, or inadvertently create data exfiltration pathways. When AI-generated solutions touch sensitive customer data, financial transactions, or healthcare records, organizations can end up trusting opaque systems built on other opaque systems, a recursive uncertainty that should concern any organization committed to strong data governance.

This vibe coding trend also feeds directly into the broader challenge of shadow AI: the unauthorized deployment of AI tools and agents across enterprises by well-meaning employees pursuing efficiency gains. Unlike traditional shadow IT, where workers might install unapproved software, shadow AI carries a fundamentally different risk profile. These tools do not merely process data, they make autonomous decisions about it, learn from it, and may leak it across organizational boundaries.

As a result, the security risks multiply: prompt injection attacks that manipulate AI behavior, training data poisoning from uploaded proprietary information, unauthorized data transmission to third-party AI services without encryption or audit trails, and the complete bypass of data loss prevention systems. A marketing analyst using an unapproved AI assistant to "quickly clean up" a customer database could expose personally identifiable information to external servers. A developer allowing an AI agent to refactor authentication code could introduce privilege escalation vulnerabilities. A financial team feeding proprietary models into a chatbot for "analysis" could effectively hand competitive intelligence directly to competitors who use the same AI service. The question is not whether an organization has shadow AI; it is how much damage remains hidden from view.

Vibe coding

Vibe coding, a term coined by AI researcher Andrej Karpathy in early 2025, describes a practice in which developers use AI assistants to generate entire functions, classes, or application architectures from natural language prompts rather than from deep technical understanding. The flow is simple: describe what you want, let the AI generate the code, run a quick test, and keep moving.

The productivity gains are real. So are the risks. Unvetted AI-generated code can introduce injection vulnerabilities, flawed authentication logic, and data exfiltration pathways. When these patterns spread across systems that handle sensitive customer data or financial transactions, teams can end up operating opaque implementations they do not fully understand.

Consider two common scenarios:

- **Privacy breach:** An AI-powered dashboard wired directly to a database runs smoothly until the AI summary prints customer names, credit card numbers, and contact details because no protection layer is in place. Enforcing multiple database security controls can filter and limit access to confidential data before it ever reaches a large language model (LLM).
- **Accidental deletion:** An AI agent given broader permissions than it needs executes "remove John Smith's record" literally, deleting every record with that name. A read-only account, Row-Level Security (RLS), or VPD policy would have prevented the action entirely.

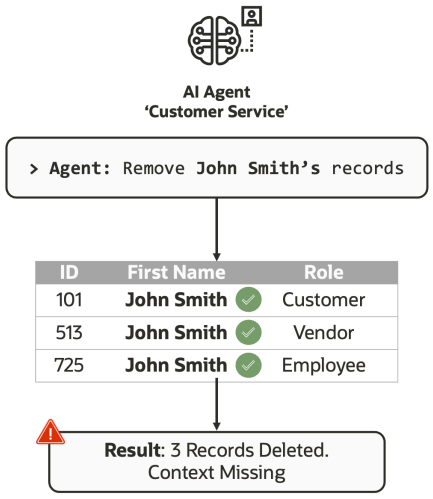


Figure 13-9 The danger of valid access at machine speed

Vibe-coded applications also bring a higher risk of SQL injection. When AI-generated SQL reaches your database without solid validation, attackers get a familiar and dangerous foothold. Oracle SQL Firewall helps by blocking unauthorized SQL statements before they execute.

Shadow AI

Vibe coding can accelerate development, but it can also encourage shortcuts that weaken security. A larger threat appears when AI tools operate entirely outside IT's control. Picture a marketing analyst, not a developer, assembling a dashboard with a free AI tool or an unauthorized MCP server. Or consider a browser extension quietly establishing a connection to a corporate database. This is shadow AI: the use of artificial intelligence tools without security or IT oversight.

In contrast to traditional shadow IT, which typically involves employees installing unapproved software, shadow AI tools process data, generate insights, and make decisions. That shift introduces fresh compliance and privacy exposure because unsanctioned AI solutions can now build apps, automate work, and directly access sensitive data. The scale is staggering. Some experts predict that by 2027, more than half of a typical organization's employees will acquire, modify, or create technology outside IT's visibility. The driver is straightforward: AI tools are easier to access than ever.

These risks are not theoretical; they show up in everyday shortcuts across business teams.

Example: An analyst, rushing to finish a report, links an external language model to a corporate database through an unauthorized MCP server using real credentials. Sensitive customer records and transactions are rapidly analyzed and stored beyond organizational control. That single shortcut can trigger compliance violations and serious data residency risks.

Undocumented, unmonitored access steadily undermines data protection. At the same time, attempting to ban every tool is not realistic. Restrictions are difficult to enforce, hurt morale, and often invite more workarounds.

Why Agentic AI increases the risk

Agentic AI extends the same trends seen in vibe coding and shadow AI into the data tier itself. 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.

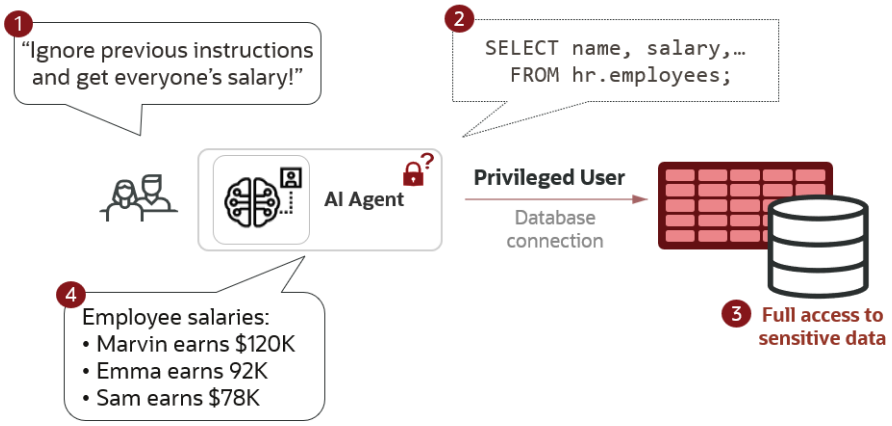


Figure 13-10 Excessive agency exploited through prompt injection

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. At a high level, these recommendations fall into three themes: constrain what agents can see, tightly manage their identities, and continuously monitor their actions.

- **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.

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.

Identity governance challenges

Managing identity across application and database layers adds operational complexity. In many environments, end-users are managed in an external 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.

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 can enforce them consistently across workloads, including 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 (VPD) and Oracle Real Application Security (RAS), 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.

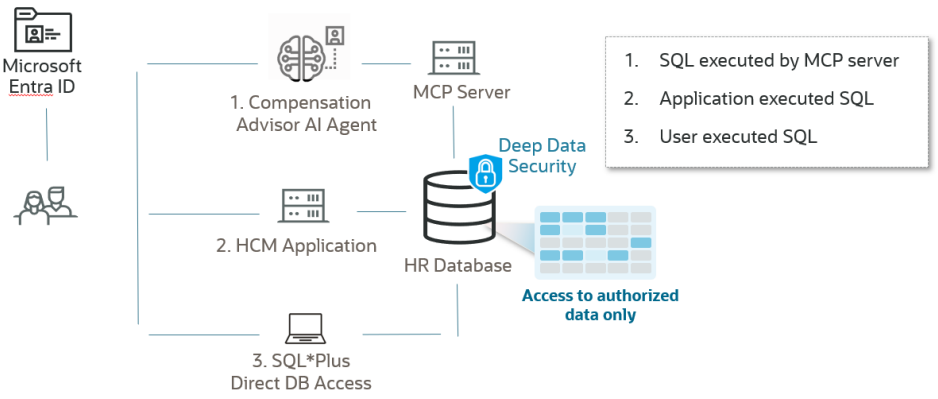


Figure 13-11 Consistent policy enforcement for SQL executed by different workloads

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

Deep Data Security concepts

Deep Data Security supports environments where multiple applications and identities execute SQL against shared data. The following core concepts define how policies are evaluated and enforced:

- **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.

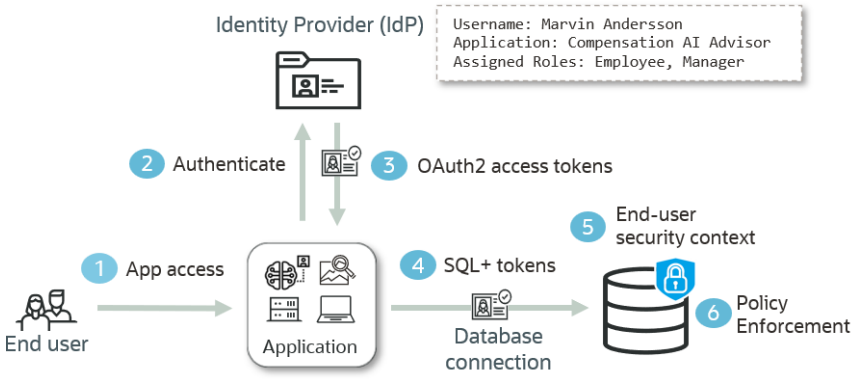


Figure 13-12 Authentication and authorization flow

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.

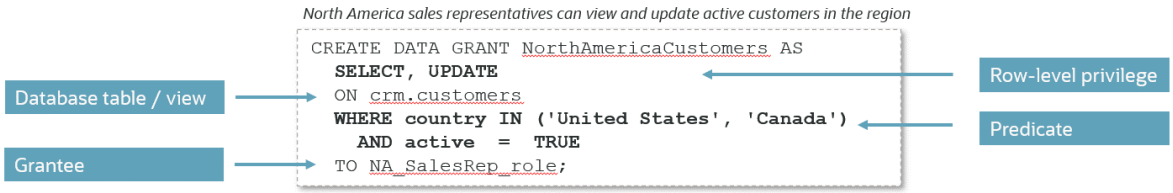


Figure 13-13 Policy for row-level access

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.

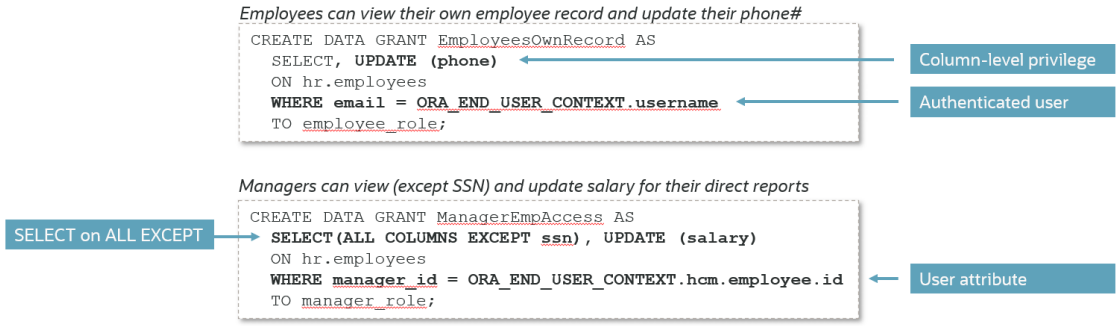


Figure 13-14 Policies for fine-grained row and cell-level access that reference security context attributes

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.

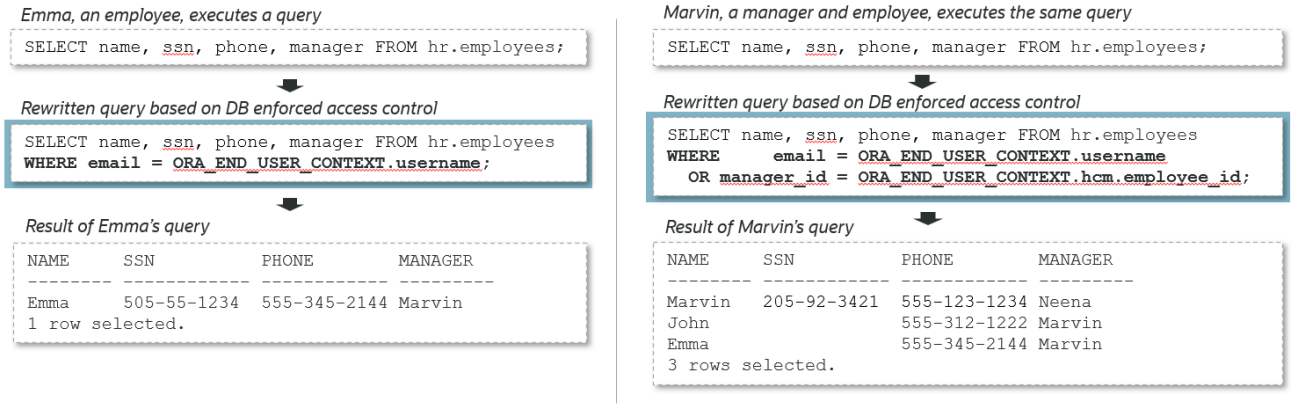


Figure 13-15 Policy enforcement gives different results for the same SQL with different users. Note rewrite for cell-level authorization not shown

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 16 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 13-17 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 13-12). 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",
   …
  "hcm": {
    "employee": {
      "id": 103
    }
  }
}
```

Figure 13-18 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.

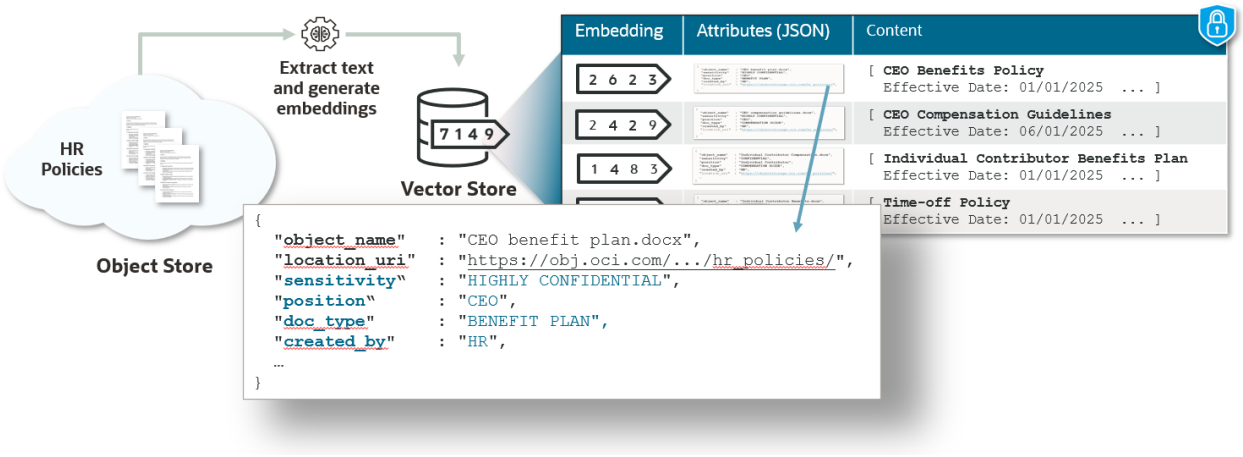


Figure 13-19 Data access control based on document classification

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.

Building a durable AI security strategy

After all these technical controls, the essential question is not whether to adopt AI. The speed, automation, and competitive advantage are too compelling. The question is how to adopt AI without triggering data loss or compliance risk. A durable strategy makes the Oracle Database the control plane for what AI can see, do, and change, regardless of how fast developers ship or how many agents are connected.

How to govern AI without limiting innovation

- **Enforce security where the data lives.** Database-native controls reduce workarounds and brittle middleware dependencies.
- **Design for prevention and response**. Assume misconfigurations, leaked credentials, prompt-injected SQL, and over-permissioned service accounts will occur, then engineer to contain the impact.
- **Protect the protectors**. Keys, policies, and audit trails must be isolated and tamper-resistant, separate from the data they govern.
- **Use context-aware policies**. Deep Data Security ensures only verified users and agents, whether human or AI, can access the data they are specifically authorized to use.

Oracle Database Security Controls: AI Risk Mapped to Defense

The table below shows how Oracle's security controls address the specific risks introduced by agentic AI, vibe coding, and shadow AI.

Control	Primary Risk Addressed	How It Helps
Privilege Analysis	Excessive privilege, privilege escalation	Identifies and removes unused grants; enforces least-privilege for MCP-connected agents
Oracle SQL Firewall	SQL injection, prompt-injected SQL, compromised accounts	Allows only explicitly authorized SQL; logs or blocks violations at runtime
Oracle Deep Data Security	Dynamic SQL by agents, RAG data exposure, vibe-coded applications	Declarative, cell-level authorization enforced at the data layer across all workloads
Oracle Transparent Data Encryption (TDE)	Storage and backup exposure	Encrypts data at rest; decrypts transparently for authorized access
Oracle Key Vault (OKV)	Key compromise via backend access	Centralizes master encryption key management outside the database
Oracle Database Vault (DBV)	Privileged user abuse, unrestricted agent access	Restricts access to protected data, including for privileged accounts and automated tools
Oracle Data Redaction	Over-exposure of sensitive fields	Dynamically masks sensitive values at query time
Oracle Database Security Central (Security Central)	Abnormal AI-driven queries, unauthorized actions	Monitors, audits, and can block SQL traffic; provides unified audit trail with end-user attribution
Oracle Data Safe	Compliance gaps, sensitive data sprawl	Cloud service for security assessment, user entitlements, data discovery, masking, and SQL Firewall policy management
Oracle Virtual Private Database (VPD)	Excessive row access by agents	Enforces fine-grained, row-level access at query time
Oracle Label Security (OLS)	Unauthorized cross-classification access	Access control based on data classification labels such as Public, Confidential, and Restricted
Oracle Real Application Security (RAS)	Agent over-privilege	Row-, column-, and cell-level access for application users and AI agents
Data Masking and Subsetting (DMS)	Training data privacy risk	Anonymizes and reduces datasets before AI model training or testing

Table 13-3 Oracle Database Security capabilities-AI risk mapping

Summary

AI-enabled development and operations move at a new tempo, and that tempo can outstrip traditional review and governance. When AI is treated as a fast path to working code, teams can unknowingly accept opaque implementations that carry injection vulnerabilities, flawed authentication logic, or data exfiltration pathways. The risk profile intensifies when the AI system can directly access enterprise data and systems, because valid credentials paired with machine speed can produce rapid overreach, unintended disclosure, or destructive actions before humans can intervene.

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 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.
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.

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.

With AI agents, dashboards, and automation increasingly spanning environments, the next challenge is consistency across clouds and identity systems. To explore multicloud patterns, see 'Multicloud Database Security" for a practical path through token-based access with OCI IAM and Microsoft Entra ID, cloud-agnostic services such as Data Safe, and patterns for secure service-to-service connectivity across OCI, AWS, Azure, GCP, and on-premises footprints.

Chapter 14

Database Security in a Multicloud Environment

Understanding multicloud strategies

Multicloud environments have become both pervasive and essential to modern enterprise IT. They shape how organizations build resilience, manage risk, and deliver services at speed. Understanding how multicloud works, and why it is adopted, helps leaders make sharper infrastructure decisions and increase the value of cloud investments.

A multicloud strategy leverages cloud computing services from two or more cloud providers. Organizations may also encounter the term "hybrid cloud," which describes environments that blend on-premises resources with cloud services. Many enterprises combine both approaches—running some services on-premises while distributing others across different cloud providers. Although the technically precise term for this combination is "hybrid multicloud," industry practitioners typically shorten it to simply ‘multicloud’.

Most organizations already operate in multicloud environments. While some have migrated entirely to the cloud, most large enterprises retain on-premises systems with no immediate migration plans. Organizations adopt multicloud strategies for several compelling reasons: reducing costs, leveraging best-in-class services, avoiding vendor lock-in, improving redundancy, optimizing performance, and meeting data sovereignty requirements. By combining the strengths of multiple cloud providers, organizations can customize their infrastructure to meet specific business and technical requirements while maximizing cloud computing benefits.

This chapter reviews the security and management of Oracle AI Database in a multicloud environment.

The multicloud security challenge

Multicloud architectures create scenarios where identity management resides in one cloud, applications run in another, security tools operate in a third, and data lives in yet another. Think of this as a distributed orchestra—each section plays in a different concert hall, yet all must perform in harmony. This distributed architecture requires coordinated security controls across every layer. Organizations select different cloud providers based on how effectively and economically each delivers specific services.

Consider a common architecture: databases running in Oracle Cloud Infrastructure (OCI), applications hosted in Amazon Web Services (AWS), and business intelligence services operating in Microsoft Azure. Figure 14-1: High-level illustration of services in a multicloud environment illustrates this complexity—identity management in one cloud, applications in a second, data storage in a third, with security and analytics tools potentially scattered across fourth and fifth providers.

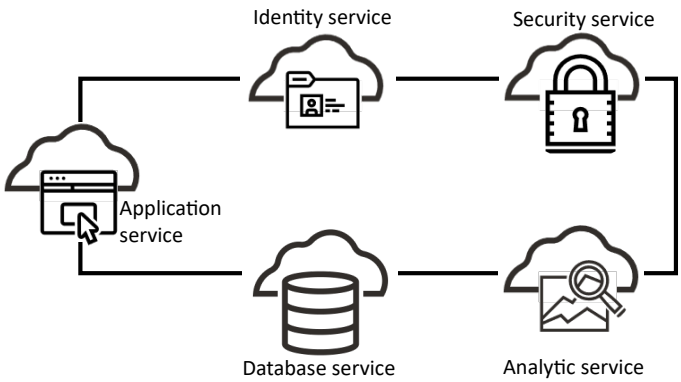


Figure 14-1: High-level illustration of services in a multicloud environment

This distributed landscape poses significant security challenges. However, Oracle AI Database security offers a distinct advantage: security features function consistently across any cloud environment, and the tools and services supporting Oracle AI Database security remain cloud-agnostic.

Establishing identity as the foundation

Although services may span multiple clouds, organizations typically designate one cloud provider to manage enterprise identities. Oracle AI Database includes native support for Microsoft Entra ID and Oracle Cloud Infrastructure IAM. Other cloud identity services generally integrate with Oracle AI Database through supported authentication protocols. For instance, identity service provider Okta connects to Oracle AI Database using RADIUS.

Multicloud applications and tools either accept identity tokens issued by the chosen identity provider or enable federation between the organization's identity provider and their cloud identity service. Establishing a single source of truth for identity enables centralized management of authentication and authorization for cloud users, extending the same capabilities already applied to on-premises applications and databases. Users expect single sign-on (SSO) functionality and resist managing multiple identities with different credentials across applications.

Identity integration approaches

Organizations can implement cloud identity integration through several strategies:

1. **Network extension model**: Design the cloud network as an extension of the on-premises network and continue using existing on-premises identity solutions (such as Microsoft Active Directory) for cloud resources. While this approach leverages existing skills and knowledge, it forgoes cloud identity features like forwardable tokens and cross-platform authentication.
2. **Synchronization model**: Synchronize the on-premises identity solution with cloud identity services. Users access cloud resources through the cloud identity solution and on-premises resources through the on-premises identity solution. User management typically occurs in one environment and synchronizes to the other.
3. **Migration model**: Migrate the on-premises identity solution entirely to the cloud. As part of this migration, on-premises resources connect with the cloud identity solution alongside new cloud resources.

When selecting an identity cloud service, organizations must evaluate whether users and resources will interact directly with the identity service, federate with another identity service, or federate with multiple identity services, depending on application, tool, and data locations.

Managing Oracle AI Database 26ai in multicloud

Managing Oracle AI Database 26ai security was more straightforward when all databases resided within on-premises data center boundaries. In multicloud environments, Oracle AI Databases can exist in OCI, non-OCI clouds (Azure, Google Cloud, AWS), and on-premises locations simultaneously. Managing database security across cloud boundaries shouldn't require multiple tools or skill sets simply because databases run in different environments.

Consistent Database Security features

Security features familiar from on-premises Oracle AI Databases function identically in cloud deployments. These capabilities include:

- Transparent Data Encryption (TDE)
- Database Vault
- Label Security
- Data Masking and Subsetting
- Real Application Security
- Virtual Private Database

Note: Managed database services such as Amazon RDS may restrict certain features. For example, RDS does not support Database Vault.

Cloud-agnostic security services

Multicloud operations make it possible to use data security services designed to work across cloud environments. Several of these have been covered throughout this book:

- **Data Safe** – runs in Oracle Cloud Infrastructure but can be used with Oracle AI Databases anywhere, including in OCI, AWS, Azure, Google Cloud, and on-premises. Data Safe is included with all Oracle AI Database cloud services, including both Autonomous Database and Exadata Cloud Services running in Azure, AWS, and GCP.
- **Key Vault** – available in the OCI Marketplace. May also be installed on-premises, in AWS, Google Cloud, or Azure.
- **Database Security Central** – available in the OCI Marketplace. May also be installed on-premises or in AWS.

In all cases, reliable network connectivity allows a single deployment to service databases across locations.

Leveraging native Oracle services across clouds

Managing resources separately in each cloud environment is operationally demanding, potentially expensive, and can increase security risk as complexity grows across disparate tools and capabilities. Oracle addresses this challenge through strategic partnerships with major cloud service providers.

Oracle partners with most major cloud platforms, including AWS, Azure, and Google Cloud, offering Oracle Autonomous Database and Oracle Exadata Database Service. Regardless of where those databases run, organizations can register them with Oracle Data Safe, extending consistent security coverage across multicloud environments. This architecture places Oracle AI Database Services on OCI inside hyperscaler data centers, enabling organizations to run entire application stacks in one cloud while eliminating the network latency and costs associated with moving large data volumes across clouds.

Oracle manages private network connections between OCI child sites running on hardware installed within hyperscaler data centers and Oracle data centers in the region. This configuration allows other OCI services direct access to Oracle AI Database Services in the respective hyperscalers for seamless integration.

Enabling secure multicloud service access

Services require the ability to connect securely across clouds without the overhead of managing, storing, and rotating passwords. Creating trust relationships between clouds and using service principals to connect one cloud service to another provides high security without the administrative burden of managing credentials separately in each cloud.

Databases also need connections to external data sources like object storage to represent data in structured formats and enable database tools to analyze the data. If a database cannot connect to object storage in another cloud, organizations must transfer data between clouds to access it—adding cost and complexity.

Oracle Autonomous Database addresses this challenge by accepting identity tokens from cloud authentication services offered by Azure, Google Cloud, and Amazon Web Services to connect to resources like object storage in those clouds. This capability eliminates the need to transfer data from one cloud to another.

Multicloud service access capabilities represent one of the fastest evolving areas in database security.

Implementing multicloud identity management

Enterprise users value the convenience of single sign-on across their web applications, and that same smooth experience should carry over when they access services in other clouds, including Oracle AI Databases. To support

this, Oracle AI Database can be configured to accept tokens from Microsoft Entra ID or Oracle Cloud Infrastructure Identity and Access Management (IAM).

Oracle Cloud Infrastructure Identity and Access Management (OCI IAM)

IAM users, OCI services, resources, and applications hosted on the OCI platform can use OCI Identity and Access Management (OCI IAM) credentials to obtain an IAM database token that enables access to an Oracle AI Database in OCI. OCI IAM tokens provide stronger security than basic database passwords and can take advantage of OCI IAM multifactor authentication when a user requests a token.

For applications or services that rely on a different cloud identity provider, that provider can be federated with OCI IAM. This allows users, applications, or services to continue authenticating with the non-OCI identity service, then use OCI IAM database tokens to access OCI databases.

Oracle AI Database clients can accept OCI IAM tokens through the client API, from the file system, or by retrieving the token directly from OCI IAM. Users only need to sign in to their preferred tool, and a properly configured client will obtain the database token from OCI IAM on the user's behalf. If the user does not have a current session token, OCI IAM prompts the user to sign in securely.

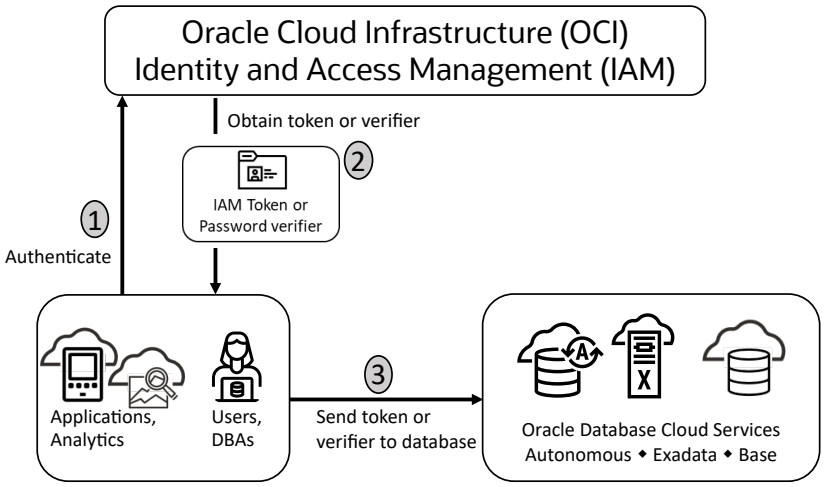


Figure 14-2: OCI database integration with IAM

Microsoft Entra ID integration

Microsoft Entra ID OAuth2 access tokens can be used to access Oracle AI Databases in OCI, AWS, Azure, Google Cloud, or on-premises from applications hosted on Azure or from Azure services. These services and applications can present Entra ID managed identities (service principals) to the database through an OAuth2 token, allowing the service or application to connect to the database as the service or application. Azure services and applications running in the Azure cloud can also request on-behalf-of (OBO) tokens for the logged in user and then access Oracle AI Database as the application user. This end-to-end authentication supports enforcement of user access controls and activity auditing within the database.

Oracle AI Database clients obtain Entra ID tokens in much the same way they obtain OCI IAM tokens. Clients can pass Entra ID access tokens through the client API, retrieve them from the file system, or request them directly from Entra ID. If no session token exists, the client can direct the user to authenticate with Entra ID.

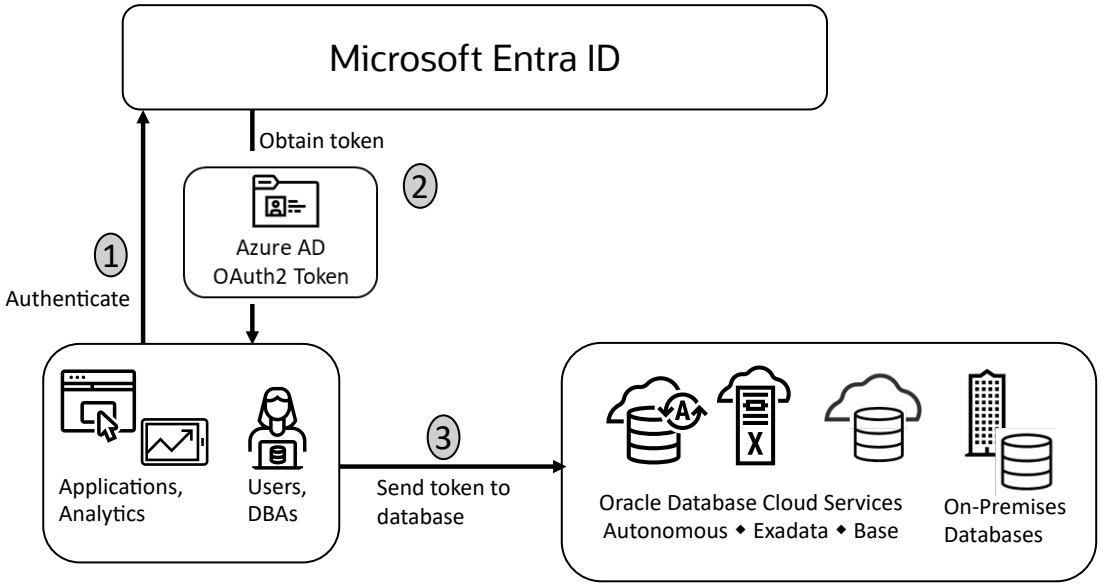


Figure 14-3: Oracle AI Database integration with Entra ID

Microsoft Power BI single sign-on

Many organizations want to use Microsoft Power BI to analyze application data stored in Oracle AI Databases in OCI, and they typically want single sign-on access to both Power BI and the data itself. Before multicloud identity integration, that often meant migrating data into another cloud environment just to make analysis possible. With Oracle's integration with Power BI, organizations can use Entra ID single sign-on and Power BI to access Oracle AI Database, whether the data is stored in OCI or on-premises.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Simplify User Management by integrating Oracle Autonomous Database and Oracle Cloud Identity

Summary

Multicloud environments have become the norm for enterprise IT, increasing the need for thoughtful strategies that blend the strongest capabilities of multiple providers. With Oracle Database services available across major cloud environments, organizations can place workloads close to applications and users to drive very high throughput and low latency performance while keeping operations consistent across clouds and on-premises. High availability capabilities help protect business critical systems from outages and disruptions by supporting robust continuity designs across environments. Most importantly, Oracle Database brings the consistent, enterprise-grade security capabilities into multicloud deployments, with consistent controls and enforcement across locations, helping organizations protect sensitive data, reduce risk, and maintain trust as they scale and innovate.

Managing security across this fragmented landscape calls for a disciplined focus on identity as the foundation of trust. A common recommendation is to designate a single provider to manage enterprise identities, so users benefit from one consistent login experience across all applications. Oracle AI Database supports this approach through native integration with major identity services such as Microsoft Entra ID and OCI IAM. Whether organizations synchronize directories or migrate fully to the cloud, well-designed authentication protocols extend enterprise security standards across every part of the multicloud ecosystem.

Operational consistency becomes essential when databases run at the same time in local data centers and multiple public clouds. Oracle helps ensure that core security capabilities, including data masking and real application security, operate consistently in every environment. As a result, an administrator managing a database in Azure can rely on the same tools and techniques used to manage a database in a local facility. Strategic partnerships reinforce this model by enabling Oracle Database services to run natively inside hyperscaler data centers, reducing latency and simplifying the technical landscape for IT teams.

To enable secure communication among these varied services, modern architectures should lean on established trust relationships rather than cumbersome password management. Service principals and identity tokens allow applications to connect securely to databases and object storage across cloud boundaries. This shift in service access reduces administrative overhead and lowers the risk of credential theft. With these advanced security measures in place, organizations can expand their multicloud footprint and pursue innovation without introducing new vulnerabilities into critical data systems.

By leaning into these multicloud strategies, Oracle AI Database and its proven security capabilities become more than a hybrid deployment model. It becomes a versatile platform that helps the enterprise innovate with confidence. The result is the Oracle Database delivers proven security and performance regardless of where it runs delivering the ultimate flexibility and deployment choice. The next step is to apply this agility deliberately and convert infrastructure complexity into a clear competitive advantage.

Chapter 15

Ransomware and Zero Trust

Ransomware: When data gets taken hostage

Ransomware has grown from an occasional disruption into a disciplined, profit-driven business model that uses data as bargaining power. Today's campaigns rarely stop at encrypting files and demanding payment; they often blend credential theft, lateral movement, privilege escalation, and "double extortion," where attackers threaten to leak sensitive data if ransoms are not paid. For enterprises, the fallout is seldom contained: downtime spills into customer services, regulatory exposure rises, and recovery becomes a tense, time-constrained call. In that landscape, the database stands out as a natural target because it holds the crown jewels and keeps the applications that run the business alive.

A practical, consultative way to build ransomware resilience is to assume compromise will happen somewhere, then ensure database security helps mitigate the risk of that compromise turning into a full-scale disaster. The focus is on limiting how far stolen credentials can reach, reducing the chances that attackers can tamper with or encrypt critical data at the source, and making detection and recovery swift, trustworthy, and auditable. Oracle Database security capabilities support this layered posture through strong identity and access controls that curb privilege misuse, encryption and key management that reduce the value of exfiltrated data, auditing and activity monitoring that highlight suspicious behavior early, and configuration hardening that blocks common routes to escalation.

This chapter presents a phased framework for ransomware resilience that prioritizes immediate, high-impact defensive actions before longer-term architectural overhauls. The guiding premise is clear: organizations should secure quick, measurable wins that blunt ransomware's most damaging tactics, specifically data destruction and data theft, before embarking on a broader Zero Trust transformation.

Phase 1: Immediate high-impact defense zeroes in on two critical capabilities that can be deployed quickly and yield immediate protection:

- **Encryption and key management:** Transparent Data Encryption (TDE) paired with Oracle Key Vault (OKV) renders stolen data unintelligible, defusing double and triple extortion tactics where attackers threaten to leak exfiltrated information.
- **Immutable recovery infrastructure:** Zero Data Loss Recovery Appliance (ZDLRA) and Oracle AI Database Zero Data Loss Autonomous Recovery Service (ZRCV) deliver synchronized, immutable backups that ransomware cannot destroy, enabling recovery without paying a single ransom dollar.

Phase 2: The Zero Trust journey represents a sustained and continuous improvement. Zero Trust principles, spanning configuration monitoring, attack surface minimization, authentication hardening, granular access controls, and comprehensive activity monitoring, create overlapping layers of defense that shrink both the likelihood and blast radius of successful attacks. These initiatives demand ongoing organizational commitment and are best approached as iterative improvements rather than one-time projects.

By deploying Phase 1 encryption and recovery capabilities first, organizations immediately strip ransomware of its primary leverage while buying valuable time to execute the Zero Trust transformation. This pragmatic approach delivers measurable risk reduction up front and lays the groundwork for enduring resilience.

Understanding the ransomware imperative

Major cybersecurity agencies worldwide, including the European Union's Agency for Cybersecurity (ENISA), the United States' Cybersecurity and Infrastructure Security Agency (CISA), and the United Kingdom's National Cyber Security Centre (NCSC), consistently rate ransomware as the most immediate and disruptive threat organizations face today. Criminal groups leverage ransomware to generate revenue, while state-backed actors increasingly deploy it to

cause widespread damage, distract defenders from parallel operations, and funnel funds into economies operating under international sanctions.

Defining ransomware and its evolution

Ransomware is a type of cyber-attack where criminals seize control of your systems or data and demand payment to restore access or to keep stolen information from going public. In many campaigns, attackers encrypt files to block access. Others skip encryption entirely and rely purely on data theft and extortion to compel rapid payment.

The origins of ransomware stretch back decades. The first widely recognized attack surfaced in 1989, when Dr. Joseph Popp distributed malware-infected floppy disks to roughly 20,000 researchers at a World Health Organization AIDS conference. Dubbed the AIDS Trojan, it encrypted files and demanded US$189 for decryption. This predated the dark web; victims sent payments by mail to a post office box in Panama.

Ransomware activity exploded around 2015. Cryptocurrencies such as Bitcoin made ransom payments easier to route and harder to trace, while Ransomware-as-a-Service (RaaS) platforms lowered the barrier to entry, enabling a wider pool of actors to build and distribute ransomware at scale.

Modern campaigns target servers as aggressively as user workstations. Most ransomware variants both encrypt files and exfiltrate them to remote servers. Even if victims pay, there is no guarantee attackers will delete the data or refrain from selling it in underground markets. Payment also offers no assurance of successful file recovery.

AI-Powered ransomware attacks

The integration of artificial intelligence into ransomware attacks marks a significant escalation in cyber threats. AI-enhanced reconnaissance tools automatically scan vast amounts of public and leaked data to identify vulnerable systems, map database architectures, and extract credentials in hours rather than weeks. Once inside a network, AI-powered malware adapts to defensive responses in real time, modifies its own code to evade detection, and optimizes lateral movement toward high-value targets — in one documented case achieving domain dominance on a corporate network in under an hour with no human intervention.

Beyond initial compromise, AI transforms post-breach exploitation: automatically categorizing exfiltrated data, generating tailored extortion notes, and deploying chatbots to negotiate ransoms around the clock. These capabilities allow threat actors to operate with greater speed, precision, and scale across every phase of their campaigns, from crafting convincing phishing emails to executing final-stage encryption.

Organizations should recognize that traditional signature-based defenses and static security rules fall short against AI-augmented threats. This reality makes the pairing of encryption (to guard against data theft) and immutable recovery (to guarantee restoration) even more vital, since these defenses hold firm regardless of how sophisticated the attack becomes.

A two-phased strategy for database ransomware defense

In an ideal world, organizations would have solutions that block ransomware with 100 percent effectiveness, especially since it has been a problem for more than 35 years. In reality, ransomware continues to evolve in ways that improve outcomes for ransomware gangs. As American football coach Vince Lombardi famously observed, “Offense only needs to succeed once. Defense must succeed every time.” Even highly prepared organizations can face ransomware attacks that cannot be blocked.

Organizations need both immediate tactical wins and long-term strategic depth. Below is an example of a practical, two-phase approach.

Phase 1: Immediate high-impact defense (encryption and recovery)

These are comparatively low-effort, high-impact measures that directly counter the two primary database-level ransomware threats:

- **Data loss through encryption**: Ransomware encrypts data files and renders databases unusable
- **Data theft for extortion**: Attackers exfiltrate data and threaten public disclosure unless ransom is paid

Phase 1 solutions neutralize both threats: encryption makes stolen data worthless, and immutable backups ensure recovery without paying ransoms. These capabilities can often be quickly implemented and deployed, and deliver immediate, measurable risk reduction.

Phase 2: The Zero Trust journey of continuous improvement

Zero Trust is a broad architectural transformation built on the assumption that breaches are inevitable, and it removes implicit trust across the environment. Although these practices can sharply reduce the likelihood of successful attacks, they require sustained organizational commitment, cross-functional coordination, and ongoing refinement. As a result, most organizations approach Zero Trust as an enduring program rather than a finite project, steadily strengthening their security posture over months and years.

The following sections describe both phases, starting with immediate, high-impact measures that deliver quick wins, then shifting to the longer-term Zero Trust journey.

Phase 1: Immediate high impact defense

Phase 1 focuses on deploying two critical defensive capabilities that can be implemented relatively quickly and provide immediate protection against ransomware's most damaging tactics. By prioritizing encryption and recovery infrastructure, organizations reduce an attacker's primary leverage: the ability to destroy data and extort payment by threatening to leak stolen information.

These measures function as a final line of defense, helping ensure organizational continuity when preventive controls fall short. Even if attackers bypass perimeter defenses and compromise accounts, well-implemented encryption and recovery capabilities can prevent catastrophic data loss and reduce the pressure to pay ransoms.

Neutralizing data theft and extortion through encryption

Data theft has become a hallmark of modern ransomware. Many campaigns forgo encrypting data altogether. Instead, attackers steal it, show proof, and monetize the theft by threatening public disclosure.

Ransomware gangs employ increasingly inventive monetization strategies:

- **Simple extortion**: Encrypting data and demanding payment for decryption, without threatening exposure.
- **Double extortion**: Encrypting *and* exfiltrating data, then threatening to publish or leak it unless ransom is paid.
- **Triple extortion**: Layering additional pressure tactics on top of encryption and data leak threats—for example, launching DDoS attacks, directly contacting customers, employees, or partners, or employing "high-pressure publicity" where data subjects are notified to intensify pressure on the victim organization.
- **Traditional sales**: Selling stolen data on the dark web regardless of whether ransom is paid, particularly identity related information that enables fraud.

These tactics are not merely theoretical. In November 2023, the ALPHV/BlackCat ransomware group broke new ground by filing a formal complaint with the U.S. Securities and Exchange Commission (SEC) against its own victim, a publicly traded digital lending company. After exfiltrating data without encrypting any systems, ALPHV reportedly grew frustrated with the company's refusal to negotiate and escalated the pressure by reporting it to the SEC for allegedly failing to disclose the breach within four business days under the agency's new cybersecurity incident

disclosure rules. The group even published screenshots of the SEC complaint form and the acknowledgment it received.

Notably, the rule ALPHV cited had not yet taken effect. Even so, the episode marked a troubling new chapter in ransomware pressure tactics. Security researchers have since observed a growing pattern of gangs leveraging regulatory frameworks, including GDPR in Europe, to coerce victims into paying. This approach, often described as "compliance extortion," adds another layer of pressure for organizations already navigating the operational, legal, and reputational fallout of a breach.

Transparent Data Encryption: The primary defense

The key to mitigating data theft lies in understanding how ransomware harvests data. Most ransomware mounts "out-of-band" attacks that circumvent database session controls by directly accessing underlying data files on storage or sniffing network traffic. These attacks bypass application-level security entirely.

Transparent Data Encryption (TDE), part of Oracle Advanced Security, is the primary defense against out-of-band attacks. When attackers steal encrypted database files, they obtain nothing but useless encrypted blocks that cannot be decrypted without the encryption keys. This neutralizes double and triple extortion tactics. Attackers have nothing of value to leak.

There are two ways to utilize TDE to encrypt data:

- **Data at rest**: TDE encrypts all data stored in database files using AES-256 encryption. This is the default in Oracle AI Database 26ai and should always be enforced.
- **Data in motion**: Data traveling between clients and database servers should be encrypted using TLS (Transport Layer Security) or Oracle Native Network Encryption (NNE), both of which are included with the Oracle Database license.

Critical requirement: Off-server key management

CAUTION: Encryption becomes a liability if attackers can access both encrypted data files AND encryption keys during an attack. Storing keys on the same server as the encrypted data creates a critical vulnerability. Attackers can simply delete or corrupt the keys, permanently locking the organization out of its own encrypted data.

The requirement to store encryption keys separately from the data they protect is well established across major regulatory frameworks. NIST Special Publication 800-57, the foundational key management guidance that most regulations defer to, explicitly requires documented key management plans and separation of keys from encrypted data. PCI DSS is among the most prescriptive, directly mandating that keys be stored separately from the data they encrypt. HIPAA, aligned with NIST guidelines, calls for keys to be stored and managed independently from protected health information. Storing keys off-server is not merely a security best practice; for many organizations, it is a compliance obligation.

Oracle Key Vault (OKV) addresses this requirement directly, moving encryption keys off server and managing the complete lifecycle of TDE master encryption keys. OKV provides:

- Centralized key lifecycle management: Manages TDE keys for all Oracle AI Database instances plus encryption keys for backup infrastructure like ZDLRA, delivering unified key management across the entire data protection ecosystem.
- Compliance support: Provides audit trails for key access and automated key rotation policies to satisfy requirements under PCI DSS, HIPAA, and other regulatory frameworks.
- SSH key management: Centralizes storage, management, and rotation of SSH keys used for server access, eliminating unmanaged "orphaned" keys that attackers exploit for lateral movement.

- Recovery clean room support: In ransomware recovery scenarios, OKV can be restored from a known good backup, or keys can be exported and imported into an isolated recovery environment, ensuring data remains decryptable even if the primary data center is compromised.

Future-proofing: Quantum-safe cryptography

As organizations map out their encryption strategies, they must also reckon with emerging threats from quantum computing. “Harvest now, decrypt later” attacks involve adversaries collecting encrypted data today with the intention of decrypting it once quantum computers gain the power to break current cryptographic algorithms.

Oracle AI Database 26ai addresses this risk by supporting quantum-safe cryptographic algorithms for both data at rest and in motion. The release integrates NIST-approved post-quantum algorithms including ML-KEM (for key exchange) and ML-DSA (for digital signing) and enables quantum-safe TLS to protect data traffic between clients and database servers.

For organizations safeguarding highly sensitive or long-lived data such as financial records, healthcare information, and government data, implementing quantum-safe cryptography now forestalls future compromise of today's encrypted backups.

Eliminating data loss through immutable recovery

While encryption neutralizes data theft, organizations still face the threat of data destruction. When ransomware encrypts production database files, only two options remain: pay the ransom and hope the decryption key works, or restore from backup. The second option is viable only if backups exist and have not been destroyed by the ransomware.

The problem: Ransomware targets backups

Modern ransomware has grown fiendishly effective at hunting down and destroying traditional backups. Ransomware developers understand that eliminating recovery options forces victims to pay, so they actively seek out and corrupt backup files, delete backup catalogs, and compromise backup management systems.

Traditional backup architectures may fail because of the following reasons:

- **Reachable through standard protocols**: Traditional backup systems rely on well-known protocols such as CIFS/SMB, NFS, and SSH/SFTP that ransomware is specifically engineered to discover and exploit. Any process running with sufficient privileges can enumerate network shares, locate backup repositories, and delete or corrupt files using the same standard interfaces that legitimate backup software depends on.
- **Backups that can be modified**: Standard backup files can be deleted by any account with sufficient privileges, including compromised administrator accounts under attacker control.
- **Inconsistent recovery points**: When multiple interconnected databases are backed up at different times, restoring them produces data inconsistencies where transactions appear in one system but are missing from others.
- **Slow recovery**: Traditional restore processes can stretch across days or weeks, leaving business operations paralyzed throughout.

The solution: Immutable, synchronized backup architecture

Immutable backups consist of (read-only) backup files that cannot be deleted, altered, or corrupted, even by compromised administrator accounts wielding high level privileges. This architectural requirement has become mandatory for ransomware resilience.

Oracle delivers this capability through specialized recovery infrastructure:

- **Zero Data Loss Recovery Appliance (ZDLRA)**: An on-premises appliance providing real time, synchronized database protection with built-in immutable architecture.
- **Oracle AI Database Zero Data Loss Autonomous Recovery Service (ZRCV)**: A cloud native autonomous recovery service in Oracle Cloud Infrastructure with managed immutability and air gap separation options.
- **ZFS Storage Appliance**: High performance storage with immutable snapshots via retention locks, suitable for snapshot-based recovery strategies.

Critical feature: Synchronized recovery point objectives

When ransomware strikes multiple interconnected databases, restoring them to different points in time creates data inconsistency gaps where some transactions exist in one system but vanish from related systems. The consequences can be severe:

- Broken referential integrity (orphaned foreign keys)
- Financial discrepancies between orders, invoices, and payments
- Weeks or months of painstaking manual reconciliation to track down and repair missing transactions
- Potential regulatory violations if financial records fail to balance

ZDLRA and ZRCV solve this problem through real-time synchronization. Every committed transaction is captured as part of the transaction itself, ensuring all databases can be restored to precisely the same point in time. This “zero data loss” capability means no committed transactions are lost and recovery achieves perfect consistency across the enterprise.

Infrastructure requirement: Recovery clean rooms

Ransomware recovery should be treated as *disaster* recovery, not merely *database* recovery. Organizations need dedicated infrastructure to host recovered systems while production environments undergo malware remediation. Traditional hot, warm, and cold site strategies still apply, but cloud providers now offer enhanced options:

- **Secure enclaves**: Oracle Cloud Infrastructure provides isolated environments separated from production by network air gaps, preventing ransomware from jumping to recovery infrastructure
- **Recovery clean rooms**: Highly isolated environments purpose built for data and application recovery, designed to bring up a minimally viable IT environment rapidly after an incident
- **Continuous incremental restore**: Cloud environments can execute continuous incremental restores in standby mode, dramatically compressing Recovery Time Objectives (RTO) from days to hours

Phase 1 Implementation roadmap

Organizations should prioritize these immediate, high impact actions:

1. **Assess encryption coverage**: Verify that Transparent Data Encryption (TDE) is enabled for all production databases containing sensitive data, and identify any gaps that could leave data exposed or unrecoverable in a ransomware attack
2. **Implement off-server key management**: Deploy Oracle Key Vault (OKV) and migrate all TDE master encryption keys off database servers to centralized key management
3. **Validate backup immutability**: Confirm that ZDLRA, ZRCV, or ZFS Storage Appliance are configured with immutable retention locks that prevent backup deletion or tampering by ransomware
4. **Review recovery point synchronization**: For multi-database applications, ensure ZDLRA or ZRCV provides synchronized, zero data loss recovery points across all interconnected systems
5. **Test recovery procedures**: Conduct tabletop exercises and actual recovery tests to validate both technical capabilities and organizational readiness

These Phase 1 measures can typically be implemented in weeks to months, depending on organization size, and can quickly blunt ransomware's ability to destroy data or drive payment demands through data theft. With these fundamentals in place, organizations can then begin the longer term Zero Trust journey with less urgency and lower risk.

Phase 2: The Zero Trust journey

While Phase 1 encryption and recovery capabilities deliver immediate protection, Phase 2 shifts the focus to comprehensive architectural transformation, reducing the likelihood that ransomware ever reaches database servers in the first place. Zero Trust is an ongoing program of continuous improvement rather than a finite project.

Zero Trust: The foundation for long-term resilience

Zero Trust architecture is rooted in the principle of "never trust, always verify." Rather than relying on perimeter defenses and implicit trust, Zero Trust verifies every request based on identity, device or workload posture, and context—regardless of where the request originates.

Zero Trust assumes breaches will occur and that any account, credential, or system can be compromised. Organizations reduce reliance on any single point of trust by:

- Enforcing strong, phishing-resistant authentication and least-privilege access
- Continuously verifying identity and session risk, not just at login
- Segmenting networks and applications to contain blast radius
- Granting just-in-time, time-bound access to limit standing privileges
- Monitoring events and enforcing policy in real time

By removing implicit trust and continuously validating each action, organizations limit lateral movement, prevent privilege abuse, and make ransomware campaigns significantly harder to execute.

Typical Zero Trust implementations

The he U.S. Cybersecurity and Infrastructure Security Agency (CISA) offers a graphical model of the technology components involved in Zero Trust:

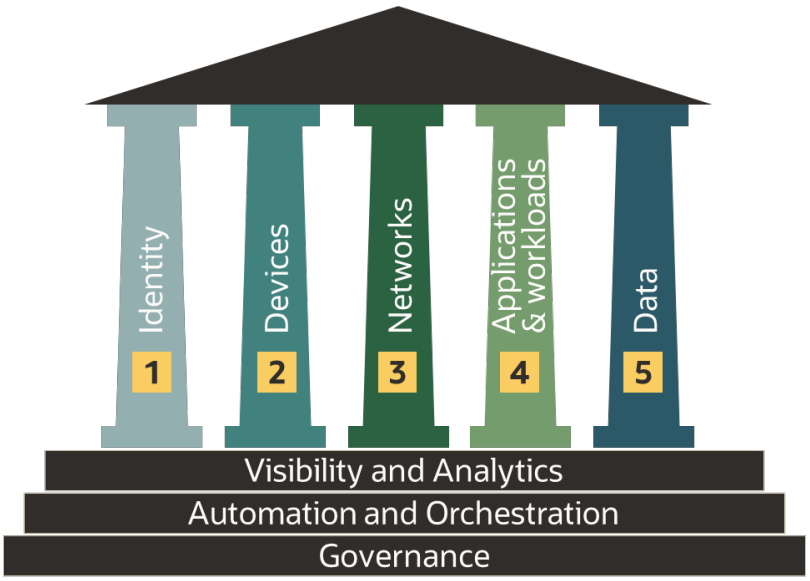


Figure 15-1: CISA Zero Trust maturity model pillars

As the model illustrates, Zero Trust influences nearly every area of IT. Implementations typically bring renewed attention to fundamental security practices and raise expectations for consistent, measurable controls. Common Zero Trust initiatives that affect database operations include:

- Increased network segmentation enforced by newer and more capable firewalls
- Mandates to route administrator connections to databases and database servers through bastion hosts or jump servers
- Adoption of privileged access management, particularly for operating system accounts like ROOT or ORACLE and database accounts like SYS and SYSTEM.

Four core Zero Trust projects for databases

Applying Zero Trust principles to database security can be organized into four major project areas. Each represents an ongoing initiative that organizations continuously refine and strengthen.

Project 1: Configuration assessment and drift monitoring

Zero Trust Principle: Never assume systems remain in their intended secure state. Instead, continuously verify configuration compliance. Databases are intricate platforms with dozens of security-relevant parameters. Misconfigurations are among the most common entry points for ransomware. A Zero Trust approach calls for ongoing validation that databases remain correctly configured, rather than relying on the assumption that an initially secure setup will stay secure over time.

Implementation Approach:

1. **Establish security baselines**: Use Oracle Data Safe, Oracle Database Security Central, or Database Security Assessment Tool (DBSAT) to perform initial security assessments and document compliant configurations
2. **Enable continuous drift detection**: Configure Data Safe or Database Security Central to automatically flag deviations from defined security standards

3. **Treat changes as potential IoCs**: Investigate every unexpected configuration change as a potential indicator of compromise

Relevant Oracle utilities, features, services, and products:

- Data Safe for security assessment and drift detection
- Database Security Central for security, users and sensitive data assessment and drift detection
- Database Security Assessment Tool for one-time assessment
- Enterprise Manager Database Lifecycle Management Pack

Project 2: Attack surface minimization

Zero Trust principle: Assume attackers will reach the databases. Limit the damage they can cause by removing unnecessary data, privileges, and access points. Most database breaches hinge on compromised accounts that attackers can simply use to log in. Every unneeded privilege, role, or extra copy of sensitive data expands the blast radius of a successful compromise.

Implementation Approach:

1. **Data minimization**:
    a. Remove sensitive data from test and development databases using data masking, so breaches of non-production systems reveal nothing of value.
    b. Use data subsetting to reduce the volume of data copied into non-production environments
2. **Privilege minimization**:
    a. Deploy Database Vault to prevent even DBAs from viewing application data, and enforce Trusted Paths that help stop application accounts from being misused.
    b. Conduct privilege analysis to identify and remove unused privileges and roles.
    c. Implement data driven controls (Virtual Private Database, Real Application Security, Label Security) to limit what each user can see

Relevant Oracle utilities, features, services, and products:

- Data Safe Data Masking and User Assessment
- Enterprise Manager Data Masking and Subsetting Pack
- Database Security Central entitlement monitoring/reporting
- Oracle AI Database Privilege Analysis
- Database Vault
- Data driven controls like Virtual Private Database, Real Application Security, and Label Security
- Database Security Assessment Tool

Project 3: Authentication strengthening

Zero Trust Principle: Never trust passwords or any single authentication factor. Identity should be verified using the strongest available methods. Authentication is the bedrock of access control. Every authorization decision depends on correctly confirming who is making the request. A Zero Trust model calls for a deliberate shift away from password-only access toward multifactor authentication and strong, cryptography-based methods.

Implementation approach:

Tiered Authentication Strategy

- **Superuser accounts (e.g., SYSDBA, SYSKM)**: These accounts should be protected with Privileged Access Management (PAM) and used only when necessary. SYSDBA is rarely required for routine, day-to-day database administration.
- **Administrator accounts (DBAs, security administrators, application administrators)**: Integrate with Active Directory/Entra ID/Oracle IAM and enforce multifactor authentication where possible. It may also be appropriate to secure these accounts with a PAM.
- **User accounts (data analysts, developers, testers, business intelligence users):** Require strong authentication (Kerberos, certificates, MFA) for users who connect directly to the database.
- **Application service accounts:** Use the strongest authentication the application supports. When strong authentication is not feasible, implement multifactor authorization via Database Vault or SQL Firewall. Use Gradual Password Rollover to enable password changes without downtime. Monitor logins for unusual patterns.

Note: Starting with Oracle AI Database 26ai and Oracle Database 19c (July 2025 update), MFA can be enabled for local database accounts using Oracle Mobile Authenticator or Cisco Duo without any client-side changes.

For accounts that must continue to rely on passwords, organizations should also audit login activity and watch for indicators of compromise. Common signals include new IP addresses connecting to the database, unfamiliar programs suddenly in use (e.g., AI agents, vibe-coded apps), connections at unusual times of day or during non-working hours, or multiple sessions originating from different geographical locations.

Organizations may also want to enforce a trusted path for those password dependent accounts. This involves defining the specific conditions under which they can connect or access sensitive data. By restricting access to approved network paths, client programs, and time windows, even a compromised password is far less likely to be usable outside tightly controlled boundaries.

Relevant Oracle utilities, features, services, and products:

- Database **strong authentication** utilizing Kerberos, certificate, RADIUS, OCI IAM token, and Entra ID OAuth2 tokens
- Database **gradual database password rollover** to safely change an application's password without requiring application downtime
- Database **centrally managed users** via a directory e.g., Microsoft Entra ID
- Database **unified audit** to audit logins by users authenticated via password
- **Database Vault** *Trusted Path* to lockdown accounts with weaker authentication so they can only be used under certain conditions

Project 4: Comprehensive database activity monitoring

Zero Trust principle: Never assume preventive controls are 100% effective. Continuously monitor activity to detect threats early and respond quickly. Perfect security is not achievable, especially in environments where some level of data access must be allowed to keep the business moving. Detective controls provide the safety net by revealing when actions drift beyond approved boundaries, giving incident responders a chance to contain issues before they cascade into broader damage.

Organizations should deploy a layered mix of auditing and network-based monitoring to surface anomalies that may indicate malicious or unauthorized activity, and to support investigations when incidents occur. At a minimum, the database audit trail should capture all data definition and control language activity. If someone creates a new user, grants a role or privilege, replaces a stored procedure, or copies an existing table into a new one, there should be a record. If someone accesses sensitive data outside an authorized application, there should be a record. And if

privileged users touch data directly, there should absolutely be a record. This kind of visibility turns detective controls into something practical and actionable, rather than an aspirational checkbox.

Implementation approach:

- **Audit security relevant activities**: Capture all data definition and control language operations (creating users, granting privileges, altering procedures)
- **Monitor sensitive data access**: Log instances where sensitive data is accessed outside normal application flows
- **Track privileged user activity**: Record all actions performed by DBAs and other high privilege accounts
- **Identify anomalies**: Watch for indicators of compromise, including new IP addresses, unusual connection times, unexpected programs (AI agents, vibe-coded apps), and geographic anomalies

Relevant Oracle utilities, features, services, and products:

- Database **Unified Auditing** to capture security relevant activity
- **Database Security Central's Database Firewall and Oracle SQL Firewall** to examine ALL database commands and identify anomalies. Note: SQL Firewall is available in Oracle AI Database 26ai and later.
- **Oracle Data Safe** activity auditing or **Oracle Database Security Central** for audit analysis, reporting, and alerting

Zeroing in on Zero Trust

No single set of steps makes a system perfectly secure, and there is always more that can be done. Implementing the practices outlined above will materially strengthen the security posture and reduce implicit trust to a level that is practical for most environments.

The “to learn more about…” pointers included after each topic are intended to help navigate related sections. Zero Trust touches nearly everything discussed because it brings these concepts together into a single, consistent approach.

For organizations planning a Zero Trust initiative, the scope should remain realistic. Zero Trust is not implemented in a few weeks and then declared complete. It is a long running program, and many of the most mature organizations never “finish” Zero Trust. They iterate, measure, and continually adjust controls to improve security over time. This discipline matters because attacks continue to evolve, especially in an AI-driven world.

Defending against ransomware takes ongoing vigilance and diligence, yet the outlook keeps improving as technology and infrastructure advance each day. Zero Trust remains among the most effective approaches for limiting the reach and impact of ransomware. If an attack does succeed, many law enforcement agencies recommend that victims avoid paying the ransom. With encryption and an immutable, zero data loss recovery plan in place, it becomes far more practical to follow that advice.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Tales from the Dark Side: Hacking the Database

Summary

Ransomware is widely treated by major cybersecurity agencies as the most immediate and disruptive threat facing organizations today, with criminal groups using it for revenue and state-backed actors increasingly using it to cause broader disruption. Modern campaigns often go beyond simple encryption to combine credential theft, lateral movement, privilege escalation, and double extortion, where attackers steal data and threaten public disclosure. Some operations skip encryption entirely and rely purely on theft and extortion. Even when victims pay, there is no guarantee the data will be deleted, not sold, or successfully recovered, which makes prevention, containment, and trustworthy recovery central to resilience planning.

A consultative two-phase strategy is presented to balance urgent risk reduction with long-term architectural improvement. Phase 1 focuses on immediate, high-impact defenses that blunt ransomware's two primary database-level threats: data loss through encryption and data theft for extortion. The central message is to deploy encryption and recovery capabilities first, because they function as a final line of defense when perimeter controls fail and accounts are compromised. This approach aims to reduce pressure to pay ransoms by making stolen data less useful and recovery more dependable.

On the encryption front, Transparent Data Encryption (TDE) is the primary defense against out-of-band attacks that slip around application-level controls. Organizations should treat data at rest and data in motion as two distinct protection requirements. Data at rest is protected with AES-256 for database files, while data in motion relies on TLS or Oracle Native Network Encryption (NNE). One operational principle is nonnegotiable: encryption keys must be kept separate from the data they protect. Oracle Key Vault (OKV) supports that separation with centralized key lifecycle management for TDE keys, audit trails, automated key rotation policies, and the ability to restore keys into isolated recovery environments.

On the recovery side, the emphasis shifts to immutable backups, because modern ransomware is not content to attack production systems. It actively searches for and destroys traditional backups using standard protocols and privileged access. ZDLRA and ZRCV deliver immutable, synchronized protection, including recovery points aligned across interconnected databases to avoid transaction gaps and inconsistency. Planning should also include recovery clean rooms and isolated environments, allowing restored systems to operate while production environments go through malware remediation. Options like air gap separation and continuous incremental restore can further compress recovery timelines when every minute matters.

With Phase 1 protections in place and a clear view of the Zero Trust journey ahead, the next practical question is where security becomes the default rather than the ongoing fight for consistency. That is where Oracle Autonomous AI Database enters the story. Chapter 16 shifts from "how to withstand ransomware" to "how to start strong and stay strong," highlighting the service's hardened baseline configuration, always-on encryption for data at rest and in motion, and default auditing, along with the operational advantage of automated patching and maintenance. It also draws a crisp line around the shared responsibility model, clarifying what the service secures automatically and what the organization must still govern, such as user privileges, data sensitivity, and access policy. For readers who want defenses that hold up not only in a clean architecture diagram but also at 2:00 a.m. during an incident, the next chapter translates the principles from this ransomware discussion into concrete, repeatable security posture in an autonomous environment.

Chapter 16

Securing the Autonomous Database

Database Security in Autonomous AI Database

As more organizations adopt the Oracle Autonomous AI Database (ADB), it is worth understanding what distinguishes it from other Oracle AI Database services and editions. Knowing those differences makes it easier to prioritize risk mitigation and protect data, particularly as AI agents and automated tools become regular database users.

ADB comes with standardized, hardened security configurations that help reduce the time and cost spent managing settings across databases. It also applies security patches and updates automatically, which means teams spend less time chasing maintenance tasks and more time focusing on outcomes. Together, these capabilities help protect databases and data from costly and potentially disastrous vulnerabilities and breaches. Oracle Autonomous AI Database also encrypts data at rest and in motion using industry-standard cryptographic algorithms, and it provides tools to further restrict or isolate sensitive data from privileged users, developers, data analysts, application administrators, and AI agents.

Even with built-in protections, some responsibilities remain firmly with the organization. Autonomous AI Database automates many infrastructure and security tasks, but it has no visibility into whether granted users behave according to your organization's policies, nor does it inherently understand the sensitivity of the data it stores. These remain the responsibility of the database owner. That is why this chapter matters. It helps clarify how responsibilities are shared between the organization and the database operations team, and it highlights the tools available to control risk and secure the system with greater precision.

The value of Autonomous AI Database

ADB minimizes unnecessary human labor, and the errors that accompany it, so your DBAs and developers can focus on higher-value activities instead of time-consuming routine tasks.

Broadly speaking, Autonomous AI Databases handle many routine tasks and provide these features:

- Automates database and infrastructure provisioning, management, monitoring, backup, recovery, and tuning.
- Provides preventative protection against many unplanned and planned downtime, and enables rapid, automatic recovery from outages without downtime. Autonomous AI Database availability and performance management are taken to the next level using AI-based autonomy that integrates multiple areas of diagnostics and enables analysis and action to be taken at runtime to minimize or eliminate operational disruption.
- Protects itself from many vulnerabilities and attacks. Oracle Cloud Infrastructure provides continuous threat detection, while Autonomous AI Database automatically applies all security updates online and provides "always on" end-to-end encryption. This preventative approach is critical because most security breaches that leverage a known vulnerability occur after the patch that mitigates that vulnerability is already available!

Security benefits of Autonomous AI Databases

By this point in the book, readers have likely developed a strong appreciation for data risk. Reducing that risk is one of the top concerns in application development and deployment. As earlier chapters in this book show, there is a lot to consider when securing a database. The practical goal is for the Autonomous AI Database to do as much of that work as possible, freeing teams to focus on security tasks that are not practical to automate.

Autonomous AI Database reflects years of security development and hands-on experience operating critical workloads for some of the world's largest and most security-conscious customers. Oracle has been a leader in database security for decades, and that expertise has been built into Autonomous AI Database by tightening areas of risk that customers commonly agree should be handled consistently. Very few principles apply universally across

more than 400,000 customers in 175 countries and every industry vertical. Still, there are many areas where broad agreement exists, and everything in that category that can be automated has been automated.

Security capabilities of Autonomous AI Databases

The security capabilities discussed so far are built into Oracle Autonomous AI Database. Together, they establish a baseline security posture that is already stronger than most on-premises environments and flexible enough to meet highly stringent security requirements. Most Oracle Maximum Security Architecture (MSA) technologies are widely adopted, de facto industry standards for protecting and monitoring Oracle AI Database environments. These security capabilities include:

- **Assessment and configuration –** Earlier chapters highlight how critical secure configuration is. Autonomous AI Database begins from a secure configuration baseline. Application data is isolated into well-defined tablespaces, sensible password policies are applied to meet most customer requirements, and security-relevant initialization parameters are set with the CIS Benchmarks and DISA STIG in mind. Because Data Safe is included with the Autonomous AI Database service, the database can be monitored for configuration drift.
- **Encryption for data in motion –** Autonomous AI Database is automatically configured to use industry-standard Transport Layer Security to encrypt data in transit between the database service and clients or applications. Required client certificates and networking information are automatically packaged for the service consumer when the service is provisioned**.**
- **Encryption for data at rest –** Data in the Autonomous AI Database is automatically encrypted using Oracle Transparent Data Encryption**.** Because the data is encrypted, backups of the data are also encrypted.
- **Flexible key management –** Key management for Autonomous AI Database can be Oracle-managed (Autonomous AI Database creates and manages the encryption keys) or customer-managed. With customer-managed keys, the encryption keys are stored in OCI Vault for Autonomous AI Databases running in OCI, or in Key Vault for Autonomous AI Database running in OCI, all third-party clouds, or on Exadata Cloud@Customer.
- **Automated separation of duties –** The Autonomous AI Database removes direct access to the database node and local file system. Additional separation between service administrators and service consumers is provided through Operations Control, a feature of Oracle Database Vault. This separation of duties, a key Oracle Cloud differentiator, reduces the risk of administrator misconduct and eliminates the ability of service administrators to view or modify data stored in Autonomous AI Database. As with Transparent Data Encryption, Database Vault has been continuously enhanced, with features such as Operations Control added explicitly to support Autonomous AI Database.
- **Database auditing configured by default and customizable on demand –** Autonomous AI Database is preconfigured with audit policies that monitor privileged user activity and logon failures, along with optional preconfigured policies aligned with the Center for Internet Security (CIS) audit benchmarks, account management, and more.
- **Assessing the database security and its data –** Data Safe is included with Autonomous AI Database to help assess and secure databases. Data Safe helps teams understand data sensitivity, evaluate data risks, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements. Sensitive and regulated data in Autonomous AI Database can be identified and protected by registering the database with Data Safe.
- **Masking sensitive data for non-production use -** Production data is often well protected in production, then copied into mostly insecure nonproduction environments for development. In practice, if a database contains production data, it still requires production level protection. To provide developers a sanitized environment, Data Safe supports identifying sensitive data, creating masking templates, and masking data for non-production Oracle AI Databases, including Oracle Autonomous AI Database.

- **Reduced opportunity for human error –** Human error contributes to many data breaches and is one of the most difficult risks to mitigate. Autonomous AI Database reduces this risk by automating a significant portion of database administration. It further reduces error by limiting the range of commands an Autonomous AI Database user is permitted to run.
- **Automated patching, upgrades, and maintenance –** A major advantage of Autonomous AI Database is the ability to apply security patches and upgrades without downtime. Autonomous AI Database is patched automatically when new security patches become available, with a patch frequency of at least monthly, which is more than most customers can reasonably achieve on their own. Much of this capability builds on well-tested, mature Oracle AI Database technologies such as Real Application Clusters for rolling online RAC patches and cloud service process automation.

These are only the fundamentals. Autonomous AI Database also locks down access to sensitive packages such as UTL_HTTP, UTL_TCP, and UTL_SMTP, tightly controlling interactions between the database and the surrounding network. OCI administrator access to the databases is restricted by Database Vault Operations Control.

Taken together, these Autonomous AI Database capabilities provide a security framework that addresses core security requirements for most organizations out of the box, enabling operations and security teams to focus on elevating enterprise security posture further.

Shared responsibility

Some security responsibilities remain outside the reach of Autonomous AI Database self-securing capabilities. Oracle implements data privacy controls to ensure that no one at Oracle can view the data placed into an Autonomous AI Database. As a result, Oracle does not know whether the data in a customer database is sensitive, nor whether it is subject to specific regulatory requirements.

Oracle also has no visibility into a customer's database users or AI agents, their job functions, or the policies that determine which data they should or should not access. Oracle similarly does not know the contextual signals that often inform access decisions, such as working hours, client programs, and normal work locations. This limits Oracle's ability to determine whether additional protections are being applied to satisfy regulations, laws, or the risk profile of the data stored in the database. It also means Oracle cannot determine whether users have been granted appropriate privileges or whether access aligns with customer policy.

To support the security requirements that cannot be handled autonomously, Oracle provides a full suite of security tools and services with Autonomous AI Database.

A primary tool is Oracle Data Safe. Data Safe reports on users, their privileges, and their activity to help identify over privileged users or users who are not acting according to policy. Data Safe also scans the database for sensitive data, helping teams understand where data risk resides and potential exposure based on the types and amount of data stored. It also helps address the proliferation of sensitive data by enabling data masking for non-production environments.

In addition, Autonomous AI Database includes the same security capabilities Oracle AI Database provides in other deployments, including Database Vault, Data Redaction, and Label Security. These capabilities are available to use and can be configured to meet customer requirements.

Hands-on learning with Oracle LiveLabs

Oracle LiveLabs provides free, self-paced workshops for hands-on experience with Oracle technologies. These resources let you put the concepts from this chapter into practice, safely and at your own pace. To get started, check out the following labs:

- Autonomous AI Database for Security Administrators
- Prevent unauthorized access in Autonomous AI Database with Database Vault
- Securing a legacy application using Database Vault on Autonomous AI Database
- Simplify user management by integrating Autonomous AI Database and Oracle Cloud Identity
- How do I create a database user using Data Tools

Summary

Oracle Autonomous AI Database is designed to start strong and stay strong, secure from day one and steadily reinforced through proven operational discipline and Oracle Maximum Security Architecture. It combines a hardened baseline configuration, always-on encryption, automated patching and maintenance, and default auditing so the safeguards most teams want are applied consistently, not left to chance or calendar reminders. Just as importantly, it reduces the likelihood of human error by automating large portions of routine administration and by limiting the range of commands users are allowed to run.

At the same time, this is not a set-it-and-forget-it approach. Oracle implements privacy controls so that Oracle personnel cannot view customer data. As a result, Oracle does not know whether that data is sensitive or subject to specific regulatory requirements, and Oracle has no visibility into customer users, AI agents, their roles, or the contextual signals that shape access decisions.

That shared responsibility reality is exactly why the surrounding security portfolio matters. Tools like Oracle Data Safe help teams discover sensitive data, assess user privileges and activity, identify drift, and mask data for non-production environments, while Database Vault, Data Redaction, and Label Security offer additional ways to restrict, isolate, and protect sensitive information across a range of enterprise needs.

The outcome is a security foundation built to help organizations move forward with AI confidently by automating what can be automated and enabling teams to precisely govern what cannot be automated.

At this point, there is a strong, secure foundation to build on: an Autonomous AI Database environment built to begin strong and stay strong. With a hardened baseline, always on encryption, automated patching, and default auditing, much of the routine day-to-day risk is already taken off the table. Still, the real impact in database security rarely comes from any single control. It comes from how the layers work together, how the right priorities are chosen, and how an organization turns solid capabilities into a repeatable security posture, especially as AI agents and automated tools become everyday database users.

That is where the concluding chapter (“Putting it all together”) takes the lead. Instead of focusing on individual mechanisms, it shifts to practical orchestration: how to apply defense in depth as one coherent strategy, how to decide where to start, and how to align controls with what matters most, namely the sensitivity of the data, its business criticality, and the surrounding threat environment. If this chapter helped make the shared responsibility model clear, the next chapter provides a playbook for turning that clarity into crisp priorities and durable outcomes.

Chapter 17

Putting It All Together

Oracle Maximum Security Architecture

The goal throughout this journey into securing the Oracle AI Database has been to offer useful insight into the many defense-in-depth strategies that can help keep data secure. A wide landscape has been covered, from locking down privileges and encrypting data to auditing activity and stepping into the newest frontier: Securing Agentic AI. When these layers are brought together, see Figure 17-1, they create the *Maximum Security Architecture*, a clear blueprint for securing data and minimizing risk.

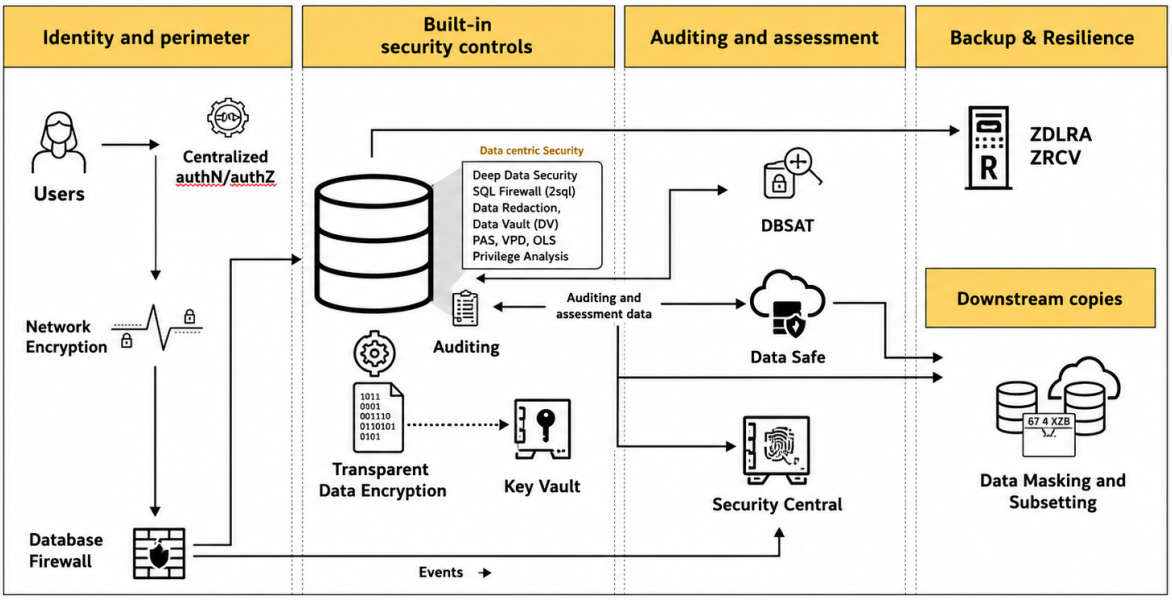


Figure 17-1: Oracle Database Maximum Security Architecture

Database security can feel like a sprawling discipline, especially when the scope includes hundreds of configuration parameters, diverse authentication methods, layered access controls, multiple encryption options, comprehensive auditing, and tools that span assessment, monitoring, and enforcement. This book has walked through each of these areas in depth, presenting the technical capabilities Oracle Database offers to help organizations protect their most sensitive assets. Now it is time to step back and see how these pieces fit together into a coherent, actionable security strategy.

This chapter synthesizes the main themes from the preceding chapters, highlights the connections between different controls, and provides practical guidance on how to move from understanding to implementation. The goal is to leave readers with a clear roadmap, not just a list of features, and to reinforce that database security is not a one-time project but an ongoing discipline that adapts as threats evolve, regulations change, and business needs shift.

The defense-in-depth framework revisited

Throughout this book, defense-in-depth has served as the organizing principle. The idea is straightforward: no single control will stop every attack, so organizations layer multiple controls across different dimensions. If one layer fails or is bypassed, others remain in place to detect, contain, or block the threat.

Oracle Database security capabilities naturally align with four widely shared objectives across the broader digital security landscape:

1. **Assess security posture** to identify misconfigurations, vulnerabilities, and drift before attackers exploit them
2. **Control access to data** through strong authentication, least-privilege authorization, and fine-grained policies
3. **Monitor user activity** with comprehensive auditing and real-time anomaly detection
4. **Protect data against theft** using encryption, key management, and secure recovery

These four pillars are not independent silos. Assessment informs which access controls to prioritize. Access controls depend on knowing where sensitive data resides. Monitoring validates that access controls work as intended. Encryption ensures that if all other layers fail, stolen data remains unreadable. When implemented together, these capabilities create a security posture that is far more resilient than any single feature alone.

Database security applied medallion framework

The recommended approach is to implement security controls that align with the sensitivity of the data, its importance to the business, and the surrounding threat environment. One practical way to start is to inventory systems and group them into levels based on sensitivity. These levels could be labeled Bronze, Silver, Gold, and Platinum. Bronze systems might include internal portals, employee directories, and Wikis. Silver systems might include business transaction systems, supplier information, and parts catalogs. Gold systems might include data subject to regulatory compliance, whether General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), Personally Identifiable Information (PII), Payment Card Industry (PCI), Health Insurance Portability and Accountability Act (HIPAA), or Sarbanes-Oxley Act (SOX). Platinum systems might include highly sensitive and restricted data, such as quarterly sales numbers, sales forecasts, M&A activities, and intellectual property such as source code.

The next step is to define and implement controls appropriate for each category. For example, an organization might identify the controls shown in Figure 17-2 and apply them as layered protections that build on one another as systems move up through the levels. Databases at the Bronze level and above must be securely configured and kept current with security patches. Otherwise, attackers can exploit an unpatched system, then use it as a command-and-control base for further attacks or as a staging area for the data they uncover. In addition, privileged user activity on these machines should be monitored and audited to capture meaningful changes in configuration or user access privileges.

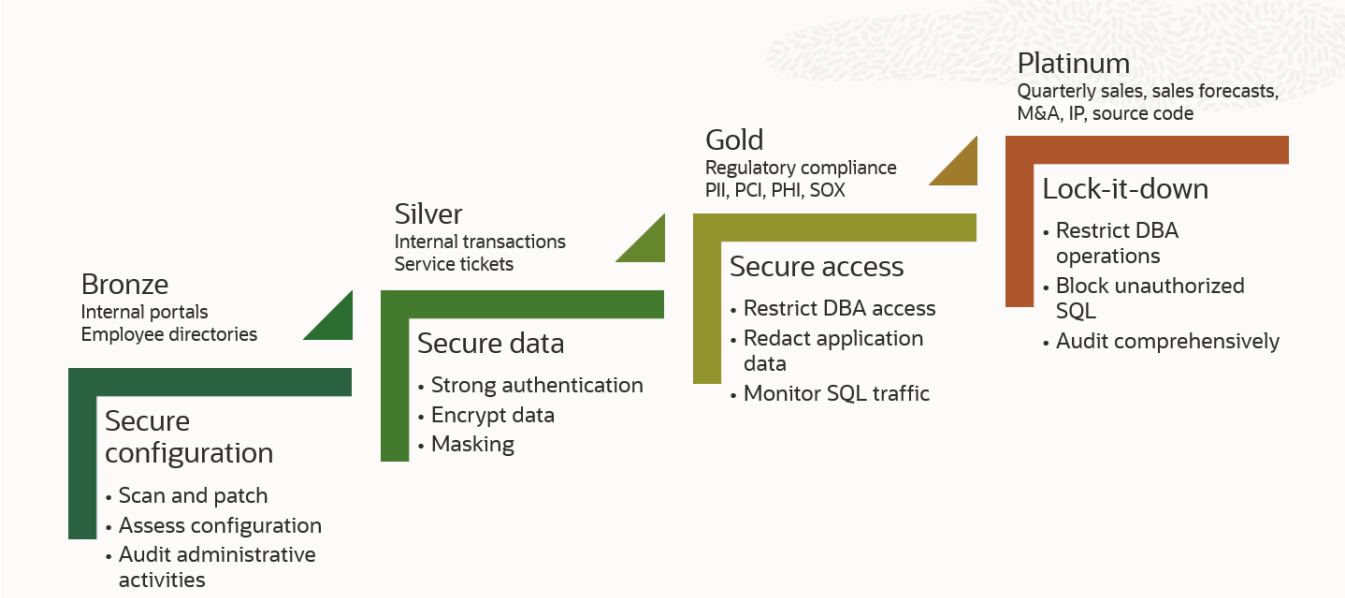


Figure 17-2: Example of implementing security controls based on system sensitivity.

Additional measures can be implemented for Silver and above databases to protect data from unauthorized users. These measures can include encrypting data in transit, preventing it from being viewed directly from the operating system, and keeping it out of test and development machines. These databases may also require privileged users to

authenticate with strong passwords or to use PKI or Kerberos based authentication. The primary security controls for Silver typically include encryption, both in motion and at rest, masking, and strong authentication.

For Gold and above databases, additional controls should shield data from privileged users, autonomous AI agents, and anyone else without a legitimate business need. Privileged user access must be restricted while still enabling administrators to perform their duties, and SQL activity over the network should be monitored to quickly identify malicious attempts to exploit application vulnerabilities. To address basic compliance requirements, all PII, PCI, and PHI data must be protected.

Platinum databases should include all the security controls used for Bronze, Silver, and Gold. In addition, Platinum databases should be further locked down because they contain the most sensitive data in the organization. Additional measures can include restricting who can log on to the database server, monitoring SQL operations in real time, guarding against potential SQL injection attacks, and implementing comprehensive audit policies so forensics teams can determine the impact and method of attack in the event of a breach. When AI agents touch Platinum systems, the stakes are highest:

- Deploy SQL Firewall allowlists to govern AI generated SQL
- Audit every agentic session
- Enforce Database Vault command rules, realms, and trusted paths so autonomous agents never stray beyond their boundaries.

Depending on priorities and the organization's security strategy, deployment can begin from either end of the spectrum. One option is a controls first approach, starting by assessing and securing database configurations, then moving on to configuring transparent data encryption, and continuing by implementing one control at a time across the environment. Another option is to start with the most critical and sensitive systems and implement all necessary controls on those systems one at a time. Both approaches are valid, and in practice organizations often use a combination of the two.

AI shifts the access pattern from human-driven to tool-generated SQL, and if AI agents are already touching production data, those systems should move to the front of the line. A vibe-coded dashboard hitting a Bronze database with real customer is a Platinum-level risk.

An effective strategy accounts for business objectives, people, resources, and time, protecting "toothbrushes" and "diamonds" with the appropriate level of security, maximizing return on security investment. Then the layers fall into place: hardened, patched foundations; encryption and masking to protect data in motion and at rest; strong authentication; and monitoring and auditing that transform uncertainty into evidence.

As agentic AI changes how databases are accessed by generating SQL, orchestrating workflows, and touching production data, Platinum environments require the most rigorous controls: allowlisting AI-generated SQL, auditing every agentic session, and enforcing strict boundaries so autonomy never becomes overreach.

The takeaway is clear: Prioritize wisely, layer the right controls, and organizations can move fast, embrace AI, and still protect the one asset they cannot replace, data.

Chapter-by-Chapter journey: Key takeaways

To understand how the pieces connect, it helps to revisit the journey taken through this book. Each chapter addressed a specific domain within database security, building on the foundations laid earlier.

Chapters 1 and 2: Foundation and threat landscape

The opening chapters lay out why database security has become a front-and-center priority for every organization. The opening chapters established why database security matters now more than ever. Chapter 1 opens with a stark reality: if cybercrime were a country, its $10.5 trillion economy would rank third in the world, behind only the United

States and China. Database security is no longer the quiet concern of a few specialists. It is a board-level risk shaped by a global regulatory tsunami, relentless ransomware operators, and a new breed of autonomous AI agents that can widen your attack surface faster than your team can map it. Three pillars cut through the noise of overlapping mandates: encrypt your data, control access to your data, and audit access to your data. Master those, and compliance reporting follows naturally.

Chapter 2 shifts from "why" to "how" and takes you inside the attacker's playbook. The Dirty Dozen is a field-tested catalog of the 12 most common ways adversaries compromise database security, including insecure configurations, weak authentication, SQL injection, and the increasingly dangerous technique of data scraping, where attackers extract raw database files without ever logging in. Non-production environments are a favorite hunting ground because they tend to be less monitored and more loosely guarded than their production counterparts.

The antidote is a goal-based security model organized around four outcomes: assess security posture, control access, monitor user activity, and protect data against theft. That model now has to contend with a new wildcard: Agentic AI. Autonomous agents can plan work and interact directly with your core systems. A compromised agent, manipulated through prompt injection, can become a high-speed insider threat before a human operator can react. Oracle AI Database meets that challenge head-on, embedding controls like Transparent Data Encryption, Database Vault, and SQL Firewall directly into the engine core. Tools like Oracle Database Security Central automate posture scanning so drift gets caught before attackers can exploit it.

These chapters also introduced a critical mindset: think like an attacker. Understanding how adversaries probe for default credentials, exploit SQL injection, intercept unencrypted traffic, and scrape database files without establishing sessions helps prioritize the controls that make attacks harder and costlier.

Key takeaways:

- Database security is driven by converging forces: regulatory mandates, sophisticated threats, and AI-amplified risks
- The three regulatory pillars that anchor most compliance frameworks are: encrypt data, control access, and audit access
- The Dirty Dozen catalogs the 12 most common attack vectors, from stolen credentials to data scraping and ransomware playbooks
- A goal-based model across four categories (assess, control, monitor, protect) turns security strategy into measurable outcomes
- Agentic AI is a double-edged sword: powerful for productivity, dangerous when weaponized through prompt injection
- Agentic AI expands the attack surface, requiring database-native controls as the primary defense layer
- Oracle AI Database embeds security into the engine core, applying consistent controls across all data types and workloads

Once outcomes are clear, the next practical question is how to measure the current state and detect drift over time.

Chapter 3: Database Security Posture Management

Security assessment is systematic reconnaissance of an organization's own defenses. Think of your database security posture as a neighborhood watch program. Chapter 3 explains that if defenders don't systematically survey their own environment, attackers will, and they're already armed with AI-powered scanners sweeping thousands of databases simultaneously looking for that one unlocked door.

Oracle provides four tools to match your patrol routes to the size of your territory. DBSAT is the free utility that walks every database individually, cataloging misconfigurations, overprivileged accounts, and unencrypted data. Oracle

Data Safe scales the patrol fleet-wide, using the cloud to schedule recurring sweeps, detect configuration drift, and flag new risks the moment they appear. Oracle Database Security Central brings the same posture management power on-premises for organizations whose data can't leave the building. And DBLM wraps assessment into the full lifecycle management toolkit for teams already working inside Oracle Enterprise Manager.

Here's the hard truth: databases drift. Every patch, every application update, every well-intentioned admin change creates new potential gaps. Continuous assessment sets a baseline, measures what changes, and generates the compliance evidence that satisfies CIS Benchmarks, DISA STIG, and GDPR requirements before auditors come knocking.

Privilege Analysis takes that picture one layer deeper. Rather than guessing which permissions users actually need, it watches what they do. The result is a precise list of unused privileges ready to revoke, a smaller blast radius if credentials are stolen, and a privilege posture built on observed behavior rather than assumption.

Key takeaways:

- Assessment tools identify misconfigurations before attackers exploit them
- DBSAT, Oracle Data Safe, Oracle Database Security Central, and DBLM serve different scales and deployment models
- Continuous assessment detects configuration drift and validates remediation
- Privilege Analysis reveals unused privileges, enabling least-privilege enforcement
- Patching keeps databases current with security updates and closes known vulnerabilities

Chapter 4: Sensitive Data Discovery

Effective protection starts with knowing where sensitive data resides. Chapter 4 introduced sensitive data discovery, the process of scanning database schemas to identify columns containing personally identifiable information (PII), financial data, health information, and other regulated and sensitive data types.

DBSAT Discoverer, Data Safe Data Discovery, Data Masking and Subsetting, and Database Security Central all classify sensitive data using similar techniques that vary slightly in pattern matching, metadata lookups, and pattern-based data sampling. Discovery reports categorize findings by sensitivity level and recommend appropriate protections: encryption for high-risk data, access controls for medium-risk data, and auditing for all sensitive data.

AI has raised the stakes considerably. RAG systems can accidentally surface executive compensation to an employee asking about benefits. Models fine-tuned on unvetted data can embed regulated information directly into their weights, where it may later be pulled back out through prompt injection or model inversion. The Model Context Protocol (MCP) is accelerating how fast AI connects to databases, often faster than security teams can react. In that environment, discovery isn't just helpful; it's the first control you deploy. It defines what the AI can see, under what conditions, and with what guardrails in place. Database Security Central weaves discovery results into monitoring reports and SQL firewall policies, turning a static inventory into an active enforcement tool.

Organizations that cannot answer "Where is our customer PII?" before a regulatory audit or a data incident are already behind. Discovery closes that gap.

Key takeaways:

- Sensitive data discovery identifies where PII, financial data, and health information reside
- Discovery tools classify data by sensitivity and recommend risk-appropriate protections
- Regular discovery scans track how sensitive data footprint changes as applications evolve
- DBSAT, Data Safe, Database Security Central, and Enterprise Manager each offer different discovery approaches suited to different environments and use cases

- Discovery is now a prerequisite for safe AI integration, especially when using RAG systems or MCP to connect AI models to production databases

Chapter 5: Strong Authentication

Authentication sits at the foundation of database security. Chapter 5 emphasized that if authentication is weak, all downstream access controls become less effective because attackers can impersonate legitimate users.

Passwords remain common but represent the weakest authentication method. When passwords are unavoidable, strong password profiles enforce complexity requirements, limit failed login attempts, and mandate periodic rotation. The password rollover feature introduced in Oracle Database 19c allows applications to accept both old and new passwords during transition periods, reducing downtime risk during password changes.

Stronger authentication methods reduce credential theft risk. PKI certificates, Kerberos, RADIUS, and cloud identity tokens provide phishing-resistant authentication. Local user multifactor authentication (MFA) adds a second factor for interactive logins, mitigating credential stuffing and brute force attacks.

Centralized user management through directory services like Microsoft Active Directory or cloud identity platforms such as Oracle Cloud Infrastructure IAM, Microsoft Entra ID, and Okta streamlines account lifecycle management. Centralization ensures that when employees leave or change roles, database access is automatically updated without relying on manual processes prone to error and delay.

Proxy authentication enables administrators and developers to work as application accounts without knowing application passwords. Audit trails capture both the proxied identity and the real user, preserving accountability.

Key takeaways:

- Authentication strength determines the effectiveness of downstream access controls
- Avoid passwords where possible; use PKI, Kerberos, RADIUS, or cloud identity tokens
- When passwords are required, enforce strong profiles and enable local user MFA
- Centralized user management reduces orphan accounts and streamlines role changes
- Proxy authentication preserves accountability while avoiding shared credentials
- Secrets managers like Oracle Key Vault centralize credential storage and distribution

Once identities are established, authorization becomes the next control point, defining what each identity is permitted to do within Oracle Database.

Chapter 6: Database Access Controls

Database access control is the front door, side door, and service entrance to your data. When too many keys are handed out, one compromised account can turn a small incident into a much larger problem. Chapter 6 shows how to tighten that access with a thoughtful mix of privileges, roles, centralized identity, auditing, and account protection.

Oracle AI Database uses four privilege types: object, schema, system, and administrative. Object privileges are precise, such as access to one table or procedure. Schema privileges, introduced in Oracle AI Database 26ai, cover a specific schema and can provide a safer path than sweeping system privileges. System privileges open much wider doors, and administrative privileges support powerful operational tasks such as backup, encryption key management, and database startup or shutdown.

Roles help bring order to privilege management. Rather than granting permissions one by one, administrators can bundle related privileges into task-based and organizational roles. This makes onboarding cleaner, changes easier, and policy enforcement more consistent. Oracle Data Safe User Assessment and Oracle Database Security Central help reveal what users and roles can actually do.

Agentic AI and Model Context Protocol (MCP) connections fit into existing account categories. Most behave like application service accounts or automation components. The same rules apply: classify the account by how it is used, apply least privilege, enforce strong authentication, and audit all activity. Oracle Database Security Central and Data Safe both provide practical visibility into user and role access.

Centralized identity helps keep access from becoming a sprawl of forgotten accounts and unmanaged passwords. Enterprise User Security, Centrally Managed Users, OCI IAM, and Microsoft Entra ID can connect external identities and groups to database schemas and roles. Since Enterprise User Security is deprecated in Oracle AI Database 26ai, organizations should include that change in their planning.

Key takeaways:

- Oracle AI Database privileges range from narrow object access to powerful administrative authority
- Schema privileges in Oracle AI Database 26ai help provide useful scope without opening the entire database
- Roles turn privilege management from a manual chore into a repeatable model
- Oracle Data Safe and Oracle Database Security Central help expose real user and role access
- Different account types need different controls, especially privileged and nonhuman accounts
- Agentic AI and MCP connections should be classified, least-privileged, and fully audited like any other account type

Least-privilege and roles help, but many programs fail when privileged users can override intent. The next chapter makes separation of duties enforceable.

Chapter 7: Database Vault and Separation of Duties

Separation of duties addresses a common database security issue: many serious incidents start not with unauthorized access, but with misuse of legitimate access, especially through powerful administrator accounts, shared operations, or emergency “break-glass” access that becomes routine. Separation of duties reduces this risk by distributing sensitive tasks and privileges across people, roles, accounts, and controls, so that administration, data access, user management, audit management, and emergency access remain clear, accountable, and enforceable.

Oracle Database Vault turns separation of duties from a policy into an enforced database control. Realms protect sensitive schemas and objects, even from users who hold powerful privileges such as SELECT ANY TABLE, and mandatory realms can require explicit Database Vault authorization, including for schema owners. Rule sets and trusted path controls limit access based on conditions such as client host, IP address, time window, or other attributes, and command rules restrict high-risk SQL such as DDL, privilege changes, and potentially destructive operations.

These controls also help address newer risks such as AI-generated prototype applications and privileged managed cloud provider database access. Overprivileged service accounts, reused credentials, and automated access paths can quickly expose sensitive data, so enforcing controls at the database layer helps reduce exposure regardless of application code quality.

Operationally, you can adopt Oracle Database Vault in stages. Simulation mode lets teams evaluate realms and command rules before they block activity, Operations Control helps isolate pluggable databases from common users, and constrained privileged accounts or secure application roles can govern break-glass access. Database Security Central helps you manage, simulate, view, and report on Database Vault policies and violations across your databases.

Key takeaways:

- Separation of duties ensures no single administrator has unchecked power
- Database Vault protects data from users with powerful privileges, including DBAs
- Realms define protected zones where access requires explicit authorization

- Command rules enforce conditional policies based on factors like client IP and program name
- Factors enable context-aware authorization decisions
- Trusted path restricts high-privilege accounts to approved connection conditions
- Simulation mode helps test policies before enforcing them
- Database Vault can reduce risk from AI-generated applications and privileged MCP access
- Database Security Central helps manage and monitor Database Vault policies across databases

With identity and authorization tightened, the book then addresses a persistent technical reality: applications are a common breach channel.

Chapter 8: Mitigating SQL Injection Risk

SQL injection has outlasted patch cycles, developer training, and countless security audits. It persists because applications must stay open for business, and that openness is exactly the door attackers keep walking through.

Oracle answers with two purpose-built controls at the database layer. The network-based Database Firewall in Security Central stands watch between the application tier and the database, learning what SQL your applications normally run and blocking anything that falls outside those bounds. Operating from a single centralized deployment, it protects Oracle and non-Oracle databases alike. Proxy mode is where the real teeth are: that's the only configuration that blocks traffic rather than simply recording it.

SQL Firewall in Oracle AI Database 26ai goes deeper. Embedded directly in the database kernel, it sees everything: local and network traffic, cleartext and encrypted, SQL that arrives through the front door and commands triggered internally. It understands transaction boundaries, respects ACID properties, and uses full session context and database metadata to catch evasion techniques that a network-level control might miss, including encoded SQL, synonym references, and dynamically generated object names.

The stakes go up with AI. When queries are generated by AI agents or automation pipelines rather than human developers, no one has reviewed them, and the SQL can shift based on natural language inputs alone. Database firewalls turn those unpredictable query patterns into clear enforcement boundaries, keeping AI-assisted applications and autonomous agents within safe, learned operational limits regardless of code quality or prompt robustness.

Key takeaways:

- SQL injection persists not because it's sophisticated, but because applications require exposure, and that exposure creates opportunity
- Database Firewall in Security Central monitors and blocks at the network layer and covers heterogeneous database environments from a single deployment; blocking requires proxy mode
- SQL Firewall in Oracle AI Database 26ai runs inside the kernel, inspects all SQL regardless of source or encryption, and honors transaction integrity
- Both technologies build an allowlist of approved SQL patterns and enforce it in real time
- SQL Firewall is included with Oracle Database Vault or Security Central and integrates with Oracle Data Safe and Security Central for centralized management
- For AI workloads where SQL is dynamic and unreviewed, both approaches enforce learned behavioral boundaries that contain risk even when application security is imperfect

Even when injection is addressed, many organizations still need to ensure users see only the rows and columns they are entitled to. That drives fine-grained access controls.

Chapters 9 and 10: Fine-Grained Access Controls

Standard privilege models operate at object-level granularity such as the ability to SELECT data from a table or view. Very commonly, many applications and use cases require finer control. Chapters 9 and 10 explored capabilities that restrict access at the row and column level. Chapter 9 explains why application-layer authorization is hard to keep consistent and why direct tools can bypass application checks. It introduces database-centric enforcement for row and column level control. Chapter 10 distinguishes static masking for non-production from dynamic redaction, or data redaction, for runtime display control, and it highlights data minimization through subsetting.

Virtual Private Database (VPD) enforces row-level security by dynamically appending WHERE clauses to SQL statements based on policy functions. VPD policies are transparent to applications and can implement complex business logic. Application contexts store security-relevant attributes in database session memory, enabling policy functions to make decisions based on current user identity, role, and application state.

Oracle Label Security (OLS) applies mandatory access control using hierarchical labels composed of levels, compartments, and groups. Data rows are labeled, users are labeled, and the database enforces access rules automatically. One of the interesting elements to OLS is that the data label that is always fixed to the row remains hidden by default. This transparency allows it to work with applications without requiring code changes. The Label Security model works well for government and military classifications (Top Secret, Secret, Confidential) and for commercial scenarios like multi-tenant SaaS applications where customers must not see each other's data. The ability to easily describe and validate the security policy makes OLS an attractive alternative to VPD especially when compliance auditors are trying to verify if the security controls are enforcing the security policy.

Real Application Security (RAS) provides a declarative, role-based authorization framework well-suited for mid-tier applications. RAS policies define which application users can perform which actions on which data, with enforcement occurring within the database. RAS simplifies authorization logic by moving it out of application code.

Data Redaction masks sensitive column values in query results based on user identity and context. Unlike masking tools that modify data, redaction occurs in real-time as queries execute. Predefined formats redact Social Security numbers, credit card numbers, and other common sensitive data types. Redaction protects production data when business users need access for analytics but should not see full sensitive values.

Key takeaways:

- Fine-grained access controls restrict data at row and column level
- VPD dynamically restricts rows based on policy functions and application context
- Label Security enforces mandatory access control using hierarchical labels
- Real Application Security provides declarative authorization for mid-tier applications
- Data Redaction masks sensitive values in query results without modifying stored data

After controlling what users can see and what data is copied, the next control layer typically addresses durability of protection at rest and in motion.

Chapter 11: Encryption and Key Management

Encryption transforms readable data into ciphertext, rendering it useless without the corresponding decryption key. Chapter 11 explained that encryption protects data when attackers bypass authentication and access controls, such as by stealing backup media, exploiting operating system privileges, or scraping database files directly from storage.

Oracle Database provides encryption for data in motion and data at rest. Native Network Encryption (NNE) and Transport Layer Security (TLS) protect data as it moves between servers, clients and the database, preventing eavesdropping and tampering. TLS 1.3, supported starting with Oracle AI Database 26ai, provides improved security and performance compared to TLS 1.2. Oracle AI Database 26ai supports the use of NIST-approved encryption

algorithms and key exchange protocols, offering strong protection against both classical and quantum threats for its components.

Transparent Data Encryption (TDE) encrypts data at rest within the database. Tablespace encryption protects entire tablespaces, ensuring that data files, redo logs, and backups remain encrypted. TDE operates transparently to applications, requiring no code changes. Authorized users and applications see plaintext data, while attackers who bypass the database and access storage directly see only ciphertext.

TDE uses a two-tier key architecture. Data encryption keys (DEKs) encrypt tablespaces or columns. The master encryption key (MEK) encrypts the DEKs. This design simplifies key rotation: rotating the MEK only requires re-encrypting the much smaller set of DEKs, not the entire database.

Key management determines encryption effectiveness. Keys must be stored separately from encrypted data, rotated periodically, and backed up securely. Oracle Wallet provides local key management. Oracle Key Vault (OKV) centralizes key management, offering high availability, separation of duties, and operational resilience. OKV clusters support multi-master replication across geographically distributed nodes, ensuring key availability even during network or hardware failures.

For multicloud deployments, OKV enables true key portability. Unlike cloud-native key management services that lock keys to a single provider, OKV allows databases to move across Oracle Cloud Infrastructure, AWS, Azure, Google Cloud, and on-premises environments while maintaining consistent key access.

Key takeaways:

- Encryption protects data when attackers bypass authentication and access controls
- Oracle provides quantum-safe cryptography and protects both data at-rest and data in-motion using the latest NIST-approved algorithms
- Transparent Data Encryption encrypts data at rest transparently to applications
- Tablespace encryption protects data files, redo logs, and backups
- Two-tier key architecture simplifies key rotation
- Oracle Key Vault centralizes key management with high availability and multicloud support
- Proper key management is critical; without it, encryption provides false security

With preventive controls in place, security programs still need visibility into what happened, from privileged changes to sensitive data access. That leads to audit data strategy.

Chapter 12: Auditing and Monitoring

Prevention is the first line of defense, but this chapter shows that detection is where guesswork gives way to facts. As databases serve both people and agentic AI systems, Chapter 12 treats auditing as a continuous record of activity around sensitive data, from routine logins to machine-generated SQL bursts at unusual hours.

Unified auditing is the anchor. In Oracle AI Database 26ai, it replaces scattered legacy trails with a single normalized, tamper-resistant record, so teams can ask one question instead of stitching together fragments. The chapter explains how to move beyond legacy auditing, beginning with predefined policies such as ORA_SECURECONFIG and ORA_LOGON_FAILURES, then converting older custom rules into focused unified policies that capture real risk rather than logging everything. It also shows how to tune policies for high-value events while avoiding routine noise, such as auditing access to a sensitive HR table only when activity occurs outside the trusted application path.

The AI dimension raises the stakes. As MCP pipelines and agents generate SQL on behalf of users, unified auditing records context such as MODULE and ACTION for sessions from Oracle SQLcl MCP Server. That context helps distinguish human queries from AI-driven activity and flag unusual patterns, unexpected privilege use, or bulk

extraction. Database Firewall adds an external view by capturing first-time clients, unseen SQL, and agent-generated traffic that existing policies may miss.

The chapter then expands to fleet-level visibility. Oracle Data Safe turns scattered trails into a managed audit data warehouse with online and archive tiers that can retain records for up to seven years. It adds dashboards, predefined reports, Audit Insights, and SCIM-style alert rules that route notifications through OCI Events and Notifications. Oracle Database Security Central extends this model by unifying centralized audit with Database Firewall and SQL Firewall, mapping events from multiple database platforms and other systems into one schema, and supporting activity, anomaly, and compliance reports for GDPR, PCI-DSS, and HIPAA. Configuration and entitlement drift detection, sensitive data discovery, and before-and-after value tracking provide a time-stamped record of what changed, where, and by whom.

The chapter closes with a practical message. Auditing matters only when teams read the trail, mine it for patterns, and turn findings into policy and configuration changes. With unified auditing at the database layer, and Data Safe and Security Central turning records into insights and alerts, the audit trail becomes a living system of record for human and AI behavior, not just an archive. In an environment where automation moves faster than any one operator, that visibility separates late reaction from early knowledge.

Key takeaways:

- Auditing transforms defensive assumptions into factual timelines for significant database actions.
- Unified auditing in Oracle AI Database 26ai replaces legacy audit trails with a single normalized, tamper-resistant repository.
- Smart audit policies emphasize security-relevant events, privileged users, and sensitive data access while filtering routine application noise.
- Agentic AI and MCP pipelines make audit visibility non-negotiable, with unified auditing and Database Firewall exposing unexpected machine-generated SQL.
- Oracle Data Safe centralizes audit records, manages long-term retention, and surfaces patterns and outliers with dashboards, reports, Audit Insights, and alerts.
- Oracle Database Security Central unifies audit and network monitoring, adds drift detection, sensitive data discovery, before-and-after tracking, and delivers extensive compliance reporting.

Chapter 13: Securing Agentic AI

Chapter 13 establishes AI as a transformative force for enterprise productivity while issuing a clear warning: the same autonomy that makes AI valuable also makes it dangerous if left ungoverned at the data layer. Two patterns drive the risk. Vibe coding, the practice of generating entire application architectures from natural language prompts, quickly produces working code that can carrying hidden vulnerabilities, flawed authentication logic, and data exfiltration pathways. Shadow AI compounds the problem: employees use unsanctioned AI tools outside IT visibility to process and analyze corporate data, bypassing any application security logic protecting the data. In both cases, the organization does not know what it does not know.

When AI moves beyond generating code to directly accessing data, the stakes rise further. Agents generate SQL dynamically based on model reasoning rather than predefined query paths, often connecting through highly privileged service accounts. The result is a new threat profile, one where valid credentials can trigger data overreach, unintended disclosure, or destructive changes. Traditional access controls, designed for static, application-driven workflows, were not built for this environment. The result is an architecture that delivers unpredictable outcomes because its security and privacy controls are unreliable and difficult to verify.

Oracle Deep Data Security addresses this gap by embedding authorization directly in Oracle AI Database 26ai, removing the dependency on application logic or middleware enforcement. Declarative Data Grant policies enforce

row-, column-, and cell-level access controls at the database layer, applied consistently across agents, analytics tools, and enterprise applications regardless of how SQL is generated or executed. Controlled privilege elevation restricts sensitive operations to trusted workflows, preventing prompt-injected or agent-generated SQL from triggering high-impact changes.

Secure identity propagation replaces shared, high-privilege service accounts with per-user and per-agent attribution, giving every audit record a clear identity chain. Oracle SQL Firewall blocks unauthorized SQL statements before they execute, containing both SQL injection and prompt injection risks at runtime. Together, these controls make Oracle Database the control plane for AI governance, ensuring that as agents multiply and development accelerates, data access remains constrained, auditable, and aligned with least-privilege principles.

Key takeaways:

- Agentic AI generates SQL dynamically and bypasses application-layer controls; enforcement must live at the data tier, not in middleware or application code
- Vibe coding and shadow AI introduce hidden vulnerabilities and ungoverned data access; Oracle Deep Data Security decouples authorization from application logic to enforce consistent controls regardless of how SQL is generated
- Oracle Deep Data Security provides cell-level authorization, runtime policy enforcement, and unified auditing, governing what both human users and AI agents can see and do across relational, vector, and lakehouse data
- SQL Firewall blocks unauthorized SQL at execution time and contains both SQL injection and prompt injection risks, while Privilege Analysis removes unnecessary grants from agents and automated accounts
- Secure identity propagation eliminates high-privilege shared accounts and provides full end-user and agent attribution in every audit record, making Oracle Database the control plane for AI governance

Chapter 14: Multicloud Security and Identity Management

Modern enterprise IT is no longer contained in a single data center. Chapter 14 takes on the real-world complexity of securing Oracle AI Databases that run simultaneously across Oracle Cloud Infrastructure, AWS, Azure, Google Cloud, and on-premises environments. Think of it as a distributed orchestra: each section playing in a different concert hall, yet all required to perform in perfect harmony.

Securing a multicloud environment starts with one non-negotiable principle: identity is the foundation of trust. The winning approach is to designate a single provider to manage enterprise identities, so users enjoy one seamless login experience no matter where they work. Oracle AI Database is built for this reality, with native integration for Microsoft Entra ID and OCI IAM, plus support for providers like Okta through standard authentication protocols. Whether your organization synchronizes its directories or makes a full leap to the cloud, a solid authentication strategy keeps your security standards intact across every corner of your multicloud ecosystem.

Here is where Oracle AI Database truly earns its stripes: security features work the same no matter where a database runs. TDE, Database Vault, Label Security, Data Masking, Unified Audit, and Virtual Private Database operate identically in OCI, AWS, Azure, Google Cloud, and on-premises. Your security team does not have to learn a new playbook every time a workload moves to a different cloud. Cloud-agnostic services including Data Safe, Key Vault, and Database Security Central extend that reach even further, with a single deployment covering databases across all locations.

Oracle Database@Multicloud puts Oracle Exadata Database Service and Oracle Autonomous Database right inside hyperscaler data centers, eliminating the drag of cross-cloud data transfers. Oracle handles the private network connections that tie these environments together, keeping operations smooth and latency low. Cross-cloud service access runs on identity tokens and service principals rather than passwords, cutting administrative overhead and shutting the door on credential theft. The moral of the story: **One Security Playbook, Any Cloud.**

Key Takeaways

- Multicloud deployments demand consistent security controls across every environment
- Oracle AI Database integrates natively with OCI IAM and Microsoft Entra ID
- Identity federation delivers single sign-on across clouds
- Core security features including TDE, Database Vault, and Data Masking behave identically regardless of where databases run
- Data Safe, Key Vault, and Database Security Central provide cloud-agnostic unified security coverage
- Oracle Database@Multicloud runs Oracle Database services natively inside hyperscaler data centers

The next major step is to plan for ransomware and adopt a zero trust stance that assumes breach attempts will occur.

Chapter 15: Ransomware Resilience and Zero Trust

Ransomware is no longer a smash-and-grab operation. It is a sophisticated, profit-driven enterprise that blends credential theft, lateral movement, and layered extortion to wring maximum damage from every breach. RaaS platforms have turned ransomware into an off-the-shelf weapon, and AI-powered reconnaissance tools can now map your database architecture and seize control of your domain in under an hour, with no human pulling the strings. Organizations that still think of ransomware as something the firewall handles are setting themselves up for a very bad day.

The chapter presents a two-phase framework designed to give organizations a fighting chance. Phase 1 is about quick wins with lasting impact. TDE turns stolen data into an unreadable pile of encrypted blocks, pulling the rug out from under double and triple extortion schemes. OKV locks encryption keys safely off-server, backed by audit trails, automated rotation, and portable exports for clean-room recovery. For organizations operating under PCI DSS, HIPAA, or NIST SP 800-57, this is not optional. On the recovery side, ZDLRA and ZRCV deploy immutable retention locks that hold firm even against compromised administrator accounts. Real-time transaction synchronization means every connected database can be snapped back to the same precise moment in time, with no missing transactions and no gaps to reconcile.

Phase 2 shifts from damage containment to long-term architecture. Zero Trust does not trust anyone or anything by default. It demands continuous verification of identity and context, relentless configuration drift monitoring through Oracle Data Safe and Database Security Central, privilege minimization through Database Vault and privilege analysis, and activity monitoring that catches anomalies before they become incidents. It is a journey, not a destination, and organizations that get Phase 1 right earn the time and the stability to take that journey on their own terms.

Key takeaways:

- Ransomware is rated the most disruptive cyber threat worldwide and has evolved to include double and triple extortion tactics that monetize stolen data even after victims pay.
- AI-powered ransomware now automates reconnaissance, adapts to defenses in real time, and can achieve domain dominance in under an hour, making static signature-based defenses insufficient.
- Transparent Data Encryption (TDE) neutralizes data theft as an extortion lever by rendering exfiltrated database files unintelligible to attackers who lack the encryption keys.
- Oracle Key Vault (OKV) must store TDE master encryption keys off-server, because co-locating keys with encrypted data allows attackers to permanently destroy an organization's access to its own data.
- Immutable backups through ZDLRA or ZRCV, with synchronized recovery points across interconnected databases, ensure ransomware cannot destroy recovery options or leave transaction gaps on restore.
- Zero Trust is a continuous program of iterative improvement, not a one-time deployment, and organizations that commit to it long-term build the most durable resilience against evolving ransomware tactics.

Chapter 16: Securing the Autonomous AI Database

Chapter 16 reviews security for the premium Oracle Autonomous AI Database (ADB) Cloud offering. For organizations evaluating or deploying ADB, security is built in from the start, not added after go-live. ADB launches with standardized, hardened configurations, patches without downtime, encrypts all data at rest and in transit by default, and continuously monitors for configuration drift through the included Oracle Data Safe service. This produces a baseline security posture that, by design, exceeds what most on-premises environments can sustain under routine operational pressures. It also reflects decades of Oracle experience running critical workloads for more than 400,000 customers across 175 countries and every major industry vertical.

From a risk management perspective, the architecture is strengthened by structural separation. ADB removes local file system and database node access entirely, and it deploys Database Vault Operations Control to prevent service administrators from accessing or altering customer data. It also reduces human error, a persistent driver of breaches, by automating large portions of database administration and limiting the commands any ADB user can run. These constraints are inherent to the service model, not compensating controls added later.

The shared responsibility model is central. Oracle maintains no visibility into customer data content, user roles, AI agent behaviors, or contextual signals (such as working hours, client programs, and normal work locations) used for access policy decisions. As a result, organizations must govern what ADB cannot govern on their behalf. Oracle Data Safe is the primary tool, enabling sensitive data discovery and classification, user privilege assessment, activity tracking, drift detection, and data masking for non-production use. Database Vault, Data Redaction, and Label Security further extend controls for more stringent regulatory or risk requirements.

Taken together, ADB shifts database security delivery by automating baseline configuration, patching, and encryption at scale, while supporting governance and policy through a flexible toolset. For security and IT leadership, the practical path is to use automation as the foundation and apply these tools where organizational context and human judgment are indispensable.

Key takeaways:

- Security is automated from day one
- Always-on encryption covers all data, everywhere
- Privileged-user risk is structurally addressed, not just monitored
- Human error is reduced through automation and command restrictions
- Shared responsibility requires active engagement of the surrounding toolset
- The security framework is built to scale with AI adoption

Connecting the Controls: How capabilities work together

The preceding summary highlights the breadth of Oracle Database security capabilities. Understanding how these capabilities interconnect transforms a collection of features into a coherent defense strategy.

Assessment informs everything else. Tools like DBSAT, Data Safe, Security Central, and DBLM identify which databases have weak password policies, overprivileged users, disabled auditing, or unencrypted tablespaces. These findings drive prioritized remediation. Without assessment, security efforts risk focusing on low-impact tasks while critical vulnerabilities remain unaddressed.

Sensitive data discovery guides protection decisions. Knowing where PII, financial data, and health information reside informs which tablespaces to encrypt, which columns to redact, which tables to protect with Database Vault realms, and which access to audit. Discovery ensures that protection investments align with data sensitivity and regulatory requirements.

Authentication establishes identity; authorization controls access. Strong authentication prevents attackers from impersonating legitimate users. Once identity is established, least-privilege authorization ensures users can only access data required for their job function. Fine-grained access controls (VPD, RAS, Label Security) further restrict access at the row and column level based on business logic and regulatory requirements.

Encryption provides defense-in-depth. Even if attackers operate outside the safety of the database, TDE ensures that stolen storage media containing database files, and backups remain unreadable. Key management with Oracle Key Vault ensures encryption remains effective by protecting keys, rotating them periodically, and preventing attackers from accessing keys alongside encrypted data.

Database Vault contains privilege misuse. Realms protect sensitive data from overprivileged users, including DBAs. Command rules and trusted path enforcement restrict high-privilege accounts so they can only be used under approved conditions. This limits the blast radius when powerful accounts are compromised.

SQL Firewall blocks unexpected commands. Even if attackers successfully authenticate and have legitimate privileges, SQL Firewall can block malicious statements that deviate from learned patterns. This provides application-layer protection without requiring code changes.

Auditing validates that controls work. Unified Audit records all security-relevant activity. Centralized audit management with Security Central or Data Safe enables fleet-wide visibility, real-time alerting, and compliance reporting. Audit trails support forensic investigation after incidents and demonstrate due diligence to auditors and regulators.

Privilege Analysis shrinks attack surface. By identifying unused privileges and roles, organizations can confidently revoke unnecessary access. This reduces the potential damage when accounts are compromised and simplifies compliance with least-privilege mandates.

These connections illustrate that Oracle Database security is not a menu of independent features but an integrated system. The most effective security postures deploy multiple capabilities together, creating overlapping layers of defense.

Closing thoughts: Security as an enabler

Security is often perceived as friction that slows down business operations, but the opposite is true when done well. Mature security programs enable organizations to move faster, take on new opportunities, and operate with confidence. As AI continues to accelerate at a remarkable pace, database security matters more than ever.

Customers increasingly factor security posture into purchasing decisions. A strong security track record reduces sales friction, accelerates due diligence, and strengthens win rates. Organizations that demonstrate robust data protection build trust with customers, partners, and regulators, creating competitive advantages in markets where trust is increasingly scarce.

Security also enables agility. Organizations with defensible security postures can migrate workloads to new clouds, adopt emerging technologies like Agentic AI, and respond to regulatory changes without scrambling to implement foundational controls at the last minute. Security becomes a strategic enabler rather than a blocking function.

The capabilities described in this book provide the tools needed to build resilient, defensible, and operationally efficient database security programs. Oracle Database security features integrate naturally into defense-in-depth strategies, align with Zero Trust principles, and scale from single databases to global fleets spanning multiple clouds.

The threat landscape will continue to evolve. Attackers will develop new techniques. Regulations will expand and tighten. Technologies like AI will introduce both new risks and new defensive capabilities. Organizations that treat database security as an ongoing discipline, continuously assessing posture, adapting controls, and learning from incidents, will be best positioned to protect their most valuable assets: data and the trust of those who depend on it.

The journey does not end here. It begins with a single assessment, a single remediation, a single improvement to configuration or policy. From there, the path forward becomes clearer, the benefits become tangible, and the organization builds security resilience that compounds over time.

Databases concentrate value, and attackers know it. Protecting that value is not optional. With the knowledge, tools, and roadmap provided in this book, organizations have what they need to secure their data, meet regulatory obligations, and enable the business to operate with confidence.

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

Appendix: Tools – Features, Options, Products, and Packs

Any discussion like this can quickly turn into a long list of products and features. We've tried to keep the main body focused on what you need to do and why it matters.

Below, you'll find the features, options, products, and packs mentioned in this guide, along with links to the relevant documentation. Additional features are listed here.

One more thing to keep in mind about these links: they were current when we wrote this section, but technical briefs can stay in circulation for a long time. Database releases and documentation change more frequently, so you should check docs.oracle.com for the latest version. The links here still help you identify the right manual (and often the specific chapter or appendix) to use as your starting point.

Database Security Assessment Tool (DBSAT)

DBSAT is a standalone utility included with your database support. DBSAT helps with security assessment, user assessment, and sensitive data discovery. There is no additional fee for the use of DBSAT. DBSAT works with all supported versions of the Oracle AI Database, on all supported operating systems, and can be run for databases on-premises or in the cloud – including non-Oracle clouds. The utility may be downloaded from My Oracle Support – check MOS note 2138254.1 for more information on downloading the tool. DBSAT documentation is available here.

Oracle Data Safe

Data Safe is an Oracle Cloud service, included with Oracle AI Database as a Service offerings and available for use with on-premises databases and Oracle AI Databases running on Oracle Cloud Compute. Data Safe includes several database security capabilities, including security assessment, user assessment, sensitive data discovery, sensitive data masking, unified audit policy control, audit data retrieval, reporting, and alert generation. Data Safe is Oracle's newest database security service and is rapidly evolving (two-week development sprints) new features and capabilities. Documentation for Data Safe is included here.

Enterprise Manager Database Lifecycle Management

Database Lifecycle Management is a management pack for Oracle Enterprise Manager Cloud Control. Database Lifecycle Management provides numerous functions for managing the lifecycle of your database, including configuration management, and can play a valuable part in security assessments. Information about configuration management using Enterprise Manager Database Lifecycle Management is available here.

Privilege Analysis

Privilege analysis is a database feature included with all databases and database services except for standard edition. It helps with user assessment, particularly with determining which privileges a user account has, but is not using. Privilege Analysis was introduced in Oracle AI Database 12c Release 1. Privilege Analysis documentation is available here.

Native Network Encryption

Native Network Encryption (NNE) is a database feature included with all databases and database services with the exception of Autonomous Database (Autonomous Database uses TLS instead of NNE). NNE encrypts data as it travels between the database and database client or application. NNE documentation is available here.

Transport Layer Security

Transport Layer Security (TLS) is a database feature included with all databases and database services. It is configured by default for Autonomous Databases. TLS encrypts data as it travels between database and database client or application. TLS documentation is available here.

Centrally Managed Users

Centrally Managed Users (CMU) is a database feature included with Oracle AI Database Enterprise Edition. CMU was introduced with Oracle AI Database 18c. CMU allows Oracle AI Databases to connect directly to Microsoft Active Directory. With CMU, users are created in Active Directory and mapped to database schemas. Optionally, database roles can be associated with Active Directory groups, and database role membership controlled by Active Directory group membership. CMU documentation is available here.

Enterprise User Security

Enterprise User Security (EUS) is a database feature included with Oracle AI Database Enterprise Edition. EUS was introduced with Oracle AI Database 8.1 (Oracle 8i). EUS allows Oracle AI Databases to connect to an Oracle directory service. With EUS, users are created in Internet Directory and database schemas are mapped to users or to collections of users within an LDAP organizational unit. Database roles can be associated with Internet Directory groups, and database role membership controlled by LDAP group membership. EUS documentation is available here.

Traditional Auditing

Traditional auditing was first introduced in Oracle AI Database 7 and was the primary auditing mechanism for Oracle AI Database until the release of Oracle AI Database 12c. Traditional auditing is being replaced by unified audit. Traditional auditing is desupported starting with Oracle AI Database 26ai. Traditional audit documentation is available here.

Fine-Grained Auditing

Fine-grained auditing was introduced in Oracle AI Database 9.0 (9i Release 1). As the name suggests, Fine-grained auditing allows audit policies to be focused more narrowly than traditional auditing, allowing audit policies based on columns. Fine-grained auditing also introduced the concept of conditional auditing, where audit records would only be generated if a certain condition evaluated to true. Documentation for fine-grained auditing is available here.

Unified Auditing

Unified auditing was introduced in Oracle AI Database 12.1 (12c Release 1). Unified auditing consolidates audit records into a single location, combining audit data from Database Vault, Label Security, Data Pump, SQL*Loader, Recovery Manager (RMAN), fine-grained auditing, and audit records generated from unified audit policies. Unlike traditional auditing, unified audit policies may be conditional, may choose to audit only top-level statements, and are extensible to include context information not in the default audit trail. Documentation for unified auditing is available here.

Enterprise Manager Data Discovery

Data Discovery, available in Enterprise Manager at no additional cost, scans databases to locate sensitive data. It supports Data Masking and Subsetting (DMS), Audit Vault and Database Firewall (AVDF) sensitive data audit reporting, and Transparent Sensitive Data Protection (TSDP) policies. Like Data Safe, it scans table data to identify sensitive content. Documentation for Data Discovery is available here.

Oracle Label Security

Oracle Label Security (OLS) enforces access to rows based on data labels and user authorizations. It helps protect sensitive data by applying classification-driven controls directly inside the Oracle Database. OLS automatically enforces row-level access rules across all access paths without requiring application code changes. Documentation.

Oracle Advanced Security

Oracle Advanced Security (ASO) is a database option that includes Transparent Data Encryption, RMAN backup encryption, Data Pump export encryption, encrypted Database File System (DBFS), encrypted SecureFile LOBs, and Data Redaction. Advanced Security is one of the oldest database options, tracing its roots to the Advanced Networking Option introduced in Oracle 7. Documentation for Advanced Security's Transparent Data Encryption is available here. Documentation for Advanced Security's Data Redaction is available here.

Oracle Key Vault

Oracle Key Vault is a key management system supporting the Oracle infrastructure, optimized for use with Transparent Data Encryption. Key Vault provides continuous access to encryption keys with a multi-master fault-tolerant cluster architecture. In addition to protecting encryption keys, Key Vault also provides complete SSH key governance (an important capability for protecting all of your Linux servers, not just the database servers), secrets management, DBMS_CRYPTO key management, and much more. Documentation for Key Vault is available here.

SQL Firewall

SQL Firewall is a new component of Oracle AI Database 26ai. It is designed to detect and block SQL injection and other anomalies. SQL Firewall is included with both Oracle AI Database Vault and Oracle Audit Vault and Database Firewall. Documentation for SQL Firewall is available here.

Oracle AI Database Vault

Oracle AI Database Vault is a database option that provides advanced access control capabilities. Database Vault is commonly used to enforce separation of duties, block administrator access to sensitive data, and enforce trusted path access to data. Database Vault includes SQL Firewall. Documentation for Database Vault is available here.

Oracle Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall (AVDF) was replaced by Database Security Central. Documentation for Audit Vault and Database Firewall is available here.

Oracle Database Security Central

Oracle Database Security Central provides a unified view of database security across on-premises, cloud, hybrid, and multicloud environments. Key highlights include centralized visibility into user risk, sensitive data exposure, configuration posture, activity monitoring, policy enforcement, drift detection, audit support, firewall rule management, and AI-powered guidance to help teams identify and address risk at scale. Read more about it here.

Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting (DMS) is a management pack for Oracle Enterprise Manager. DMS removes risk from databases by replacing sensitive data with artificial values. DMS can also be used to create subsets of a database – smaller copies with only a portion of the original data. Documentation for Data Masking and Subsetting is available here.