ORACLE
Database

Oracle Database 19c

Real Application Security

WHITE PAPER

PURPOSE STATEMENT

This document provides an overview of the Database feature Real Application Security. It is intended solely to help you assess the business benefits of this feature and to plan your I.T. projects.

DISCLAIMER

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of your application code.

TABLE OF CONTENT

INTRODUCTION

Over the past 20+ years the traditional client server model has given way to the 3-tier model as the prevalent architecture. During this transition, security controls have moved further away from the actual data itself, weakening end-to-end security of the application, and complicating the development process. This change has also complicated enforcement of privacy and compliance regulations and created gaps in security that increase the potential for data breaches and other unauthorized access. Numerous security challenges exist in the 3-tier model:

- Security models and authorization logic are application-specific, resulting in inconsistent and incomplete security enforcement models across multiple applications.
- Identity propagation and the ability to audit end user activity are compromised as operations are performed by a single user within the database. Application-specific audit becomes fragmented as the developer must create audit logs.
- Access control policies are embedded within the application logic, with each application providing its own policy infrastructure. Maintenance costs increase and policy extensions become more complex.
- Separate access control policies are required for each data entry access point — Java in the middle tier, embedded PL/SQL logic, or direct db connection. Unchecked connections may gain unrestricted access.
- Policy administrators lack complete visibility into the security model due to the lack of a common policy model and enforcement infrastructure across the application and the database tiers. Policies are embedded as procedural logic in both applications and the database.

The application security policy and its custom enforcement today can thus lead to fragile, fragmented, and vulnerable applications. When data security moves farther away from data, all stakeholders face additional challenges:

- Application Architects: Without the knowledge of application end users within the database, the access control enforcement is largely left to the application developer. As application code executes in the database as a privileged database user with full access to application data, application developers need to be very careful in preventing unauthorized access or data leakage.
- Application Security Administrators: With each application building its own security policy constructs and enforcement mechanisms, security teams struggle to verify application security policies and how they impact enterprise-wide data access policies. As application middleware or an application firewall may not cover all data access paths to the database, application-enforced security leaves sensitive data vulnerable to many attacks including application-bypass attacks or insider attacks when the attacker connects directly to the database.
- Security Auditors: As the database does not know the identity of end-users, it cannot natively audit end-user activities, leading to weaker accountability.

REAL APPLICATION SECURITY OVERVIEW

Oracle Real Application Security (RAS) introduces the next generation of application access control framework within the database enabling 3-tier and 2-tier applications to declaratively define, provision, and enforce their security requirements. Oracle RAS introduces a policy-based authorization model that recognizes application-level users, privileges, and roles within the database, and then controls access on both static and dynamic collections of records representing business objects. With built-in support for securely propagating application users' sessions to the database, Oracle RAS allows security policies on data to be expressed directly in terms of the application users, their roles and security contexts. Oracle RAS can also act as an authorization decision service to assist the application in enforcing security within the middle-tier.

Oracle RAS implements the following security design principles:

- The access control decision on data is based not on the highly privileged database user or the schema owner but the privileges of the end-user. Access control enforcement follows the principle of least privilege so that the end-user's execution context is constrained by the privileges required for his/her tasks.
- Access control policy is enforced as close to data as possible to enforce a uniform security policy that is independent of the application code or the access path through 2-tier or 3-tier applications.
- Access control policy is declarative and separated out from application procedural logic to simplify application development and security policy administration. Declarative access control policy allows new security requirements on existing applications to be defined, specified, modified, and enforced with minimal or no application code changes.
- Data access control policies support widely used patterns such as primary key/foreign key, master-detail, organization trees, multi-tenant stripping, or common data access patterns that can be parameterized.

Oracle RAS thus offers stronger security than traditional solutions within an application as it is enforced based on end-user authorization context in the database and regardless of entry points: direct, Oracle Application Express (APEX), or middleware application servers. Enforcement of data access control is optimized based on different data access patterns as the database has the complete view of the authorization policy and the executing user's session context. Oracle RAS simplifies application development with declarative policy, as it relieves writing and embedding authorization code and ensuring the safety of such code. It also simplifies policy administration with uniform security across the middle-tier and the database.

Using declarative access control policies on application data and operations, Oracle RAS enforces security close to the data and enables end-to-end security for both 3-tier and 2-tier applications.

SECURITY REQUIREMENTS FOR DEVELOPING APPLICATIONS

Applications typically first define their information model, and then enforce operational and data security based upon factors such as users, their group membership responsibilities, as well as other constraints dictated by system, environment, and corporate policies. In the following, we present some of the operational security requirements in a typical Human Resource (HR) application:

- All employees can view phone number, work address, email, job title, project assignment, and the organization tree. However, employees can update only their own cell phone, home address, and bank information. Similarly, employees can view their own salary, and government identifier such as Social Security Number (SSN).
- Managers can view only their employees' salaries and can update the salaries only during the "salary review" period. However, they can update their title and the name of their projects at any time. They can also delegate the rights to update the title and project name to their administrators.
- HR managers can view all data for all of the employees within their assigned group unless there are geographical constraints such as the EU regulations.
- Specific people from the legal department can get full access to data of a specific employee.
- Payroll report application, a 2-tier tool, can access all employees' SSN, bank information, and home address, but not be able to read the stock options granted or the employee's performance ratings.
- Internal audit teams can see audit records of who accessed or updated sensitive data.

These operational requirements lead to the following security requirements for the HR application:

- Access application data based upon the users' identity, and their group or role, e.g., manager, legal, or human-resource.
- Access data in the sensitive columns, e.g., access SSN and Salary only if authorized.
- Delegate managers' access rights to others at an individual privilege level.
- Enforce additional constraints based upon time, geography, and data to enforce organizational and government requirements.
- Limit access to only authorized data to 2-tier management and reporting tools.
- Audit end-user and administrator activities, and track any violations.

The example above is for an HR application, but similar requirements apply to commercial packaged applications, or custom in-house applications.

DEVELOPING SECURE APPLICATIONS TODAY

While securing applications requires addressing a broad range of considerations such as authentication, authorization, vulnerability scanning, firewall protection, and secure configuration, for this paper, we focus mainly on authorization. Before discussing Oracle RAS, here we summarize how data-centric applications are secured today and the limitations of such solutions.

Database bound applications can be categorized into two groups. A typical 3-tier application runs in the middle-tier, primarily implementing or using authorization services from application frameworks, and connecting to the database as a privileged schema user. On the other hand, 2-tier applications for managing the meta-data or for installing patches, run as a trusted application with full access to the data, or with enforcement coded within the database stored procedures. If we add requirements to control access by administrators, the security picture soon becomes complex as shown in Figure 1. The padlocks indicate points at which access control is enforced. Patching and Direct User Access are assumed to rely on DB security mechanisms for enforcement. For securing data, it is not sufficient to only address the security issues for 3-tier applications, but also do it for all related privileged applications irrespective of data access paths.

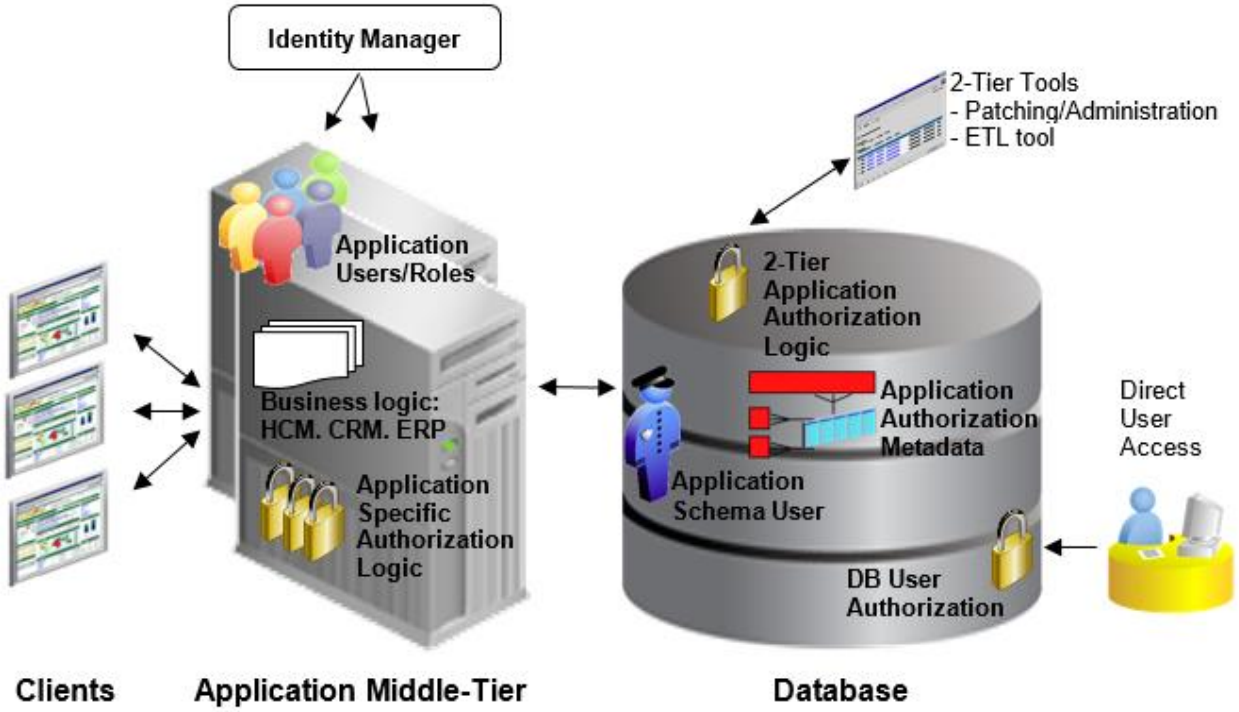

Figure 1. *Typical security enforcement in multi-tier applications*

A typical 3-tier application running in the application server, connects to the database as a privileged schema user, and then enforces appropriate access rights based upon the application users and privileges as shown in Figure 1. The existing database object privileges — select, insert, update, and delete — for database users and roles is of limited use for 3-tier applications with thousands of application users. The application architect models complex security requirements on a set of related tables where a list of applicable rows and columns represent the application objects, the subjects are application users and roles, and where the operations are application-specific. These application constructs — objects, roles, operations — are typically stored and managed in the database directly by the application, and the application developers then express and implement custom security policies using them. The access control policy is dispersed in the application code, residing partly in program code in the middle-tier and partly in stored procedures in the database. Application developers are also responsible to secure the authorization metadata as well as enforcement modules.

Here are some of the common techniques used for implementing authorization today:

- One Big Application User with Connection Sharing

As database connections are expensive to maintain at a per-user level in multi-tier applications, applications share their database connections for all application logic. To mitigate access control and accountability concerns, some applications store the end-user security context in database session using application variables, but this requires frequent switching of the security context, affecting performance and scalability.

Since the identity and security context of application users is not securely available inside the database, the application has to implement and maintain the auditing logic. Applications may use additional columns or tables to record audit data, but this auditing is not integrated with the auditing service provided by the database.

- Query Rewriting

Query rewriting mechanisms, either through application logic or Oracle's Virtual Private Database VPD) along with Secure Application Context, can help applications protect sensitive information by applying row-level access control rules on queries. However, query rewriting mechanisms require manually implementing the security policy in middleware program code or server-side stored procedures. Depending upon the application, the query rewriting rules can become very complex, making it very difficult to extend and understand. If VPD is used, it only provides a callout function hook to append a WHERE clause to filter data, but the application still has to build the infrastructure for providing information on

application user/roles, session state, and the application-level operation requesting access, as well as the application level primitives required to create the filter. Without a common infrastructure, each application has to manage its own meta-data making it difficult to understand and reuse.

The traditional databases lack the following required features for developing secure applications:

- The database does not recognize application end users and their application-level authorizations or roles, which are often provisioned in Identity Management (IDM) stores. Without application users known to the database, the database cannot support application sessions.
- Application-level access control requirements are not on database tables or views, but typically on attributes of records or fine-grained business objects such as purchase orders, employee records, Personally Identifiable Information (PII) attributes of an employee, and so on. The data for a target business object can span across many tables or views of different database schemas. In consolidated or cloud deployment of applications, the security policy is on tenant-specific data, where tenant identifier is the target attribute for the security policy.
- Lastly, application data security policy is specified based on application-level operations on these business objects such as "approve employee vacation request" or "view PII data". Traditional databases have not provided application operation based authorization primitives to specify such policies on rows and columns of database tables.

ORACLE REAL APPLICATION SECURITY MODEL

Figure 2 shows the same multi-tier application in Figure 1 but developed and deployed using the Oracle RAS security framework. Oracle Database 18 Real Application Security (RAS) provides a rich, declarative access control model that natively supports application-specific authorization primitives — users, roles, privileges — within the database. For Data Security, the RAS enabled database can make data access decisions automatically so that only relevant rows and columns are returned to the user. The application can also use RAS to find the relevant access control policies on application operations and enforce them in the middle-tier. RAS authorization results can be cached in the middle-tier query result sets to avoid database roundtrips. Oracle RAS is designed to address both the security and performance issues.

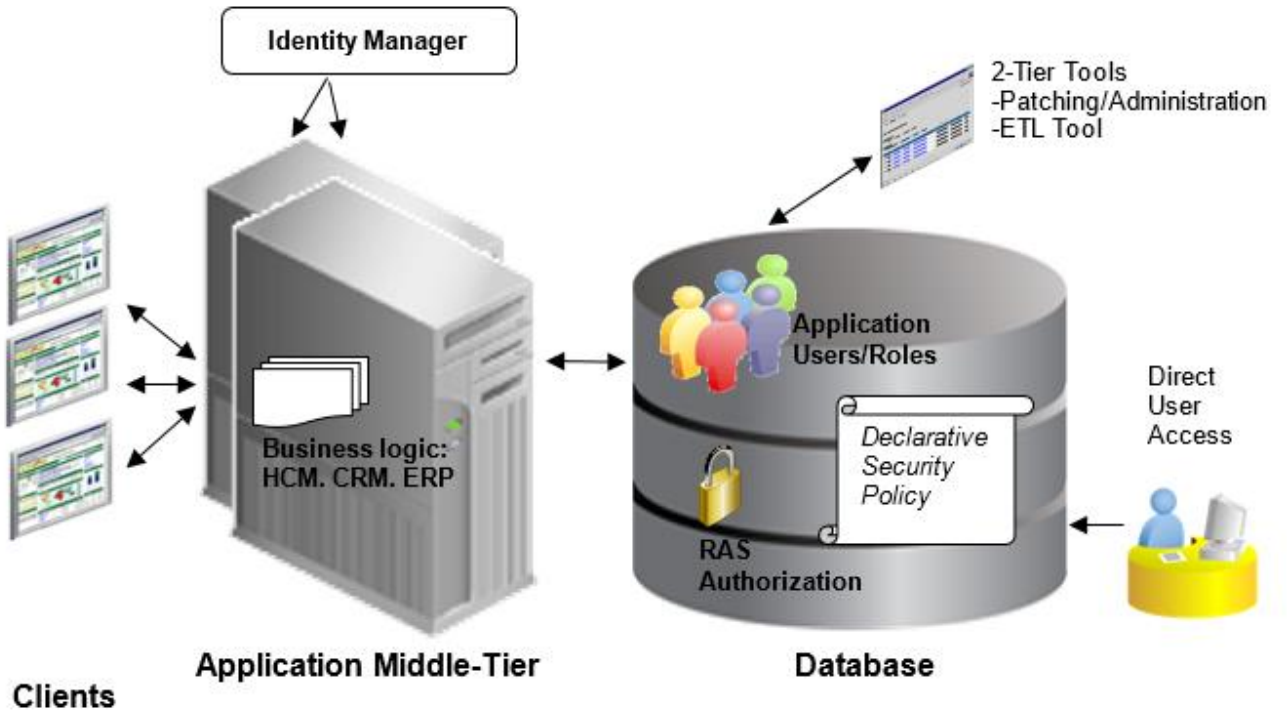

Figure 2. *Security enforcement in multi-tier applications using Oracle RAS*

The Oracle RAS model introduces the following components within the Oracle database:

- Application Users: Application end-user identities that are unified across the application and the database tiers. These users are schema-less and do not own any database objects or resources.

- Application Privileges: Named privileges to control execution of application-level operations. The operations can be on rows or columns of a database table or application artifacts such as UI artifacts representing workflow tasks or buttons and pages in a Web application.
- Application Roles: Groups of application privileges or other application or database roles. During user provisioning, these roles are assigned to application users.
- Data Realms: A collection of a logical set of rows in a table or in a group of related application tables. A Data Realm is the primary construct to specify data security policy. It represents an application-level resource or business object and is defined using a SQL predicate.
- Session Namespace Attributes: Collections of attribute-value pairs that can be used in the SQL predicate to define a data realm. Each collection is managed under an application namespace with associated access control policy.
- Application Sessions: Application users' sessions that correspond to application users' security contexts - roles and namespace attributes - in the database. These end-user sessions are created through the application tier and are natively supported in the database.
- Access Control Lists (ACL): A named list of privilege grants to users or roles. Oracle RAS ACL allows various constraints on the privilege grants such as ordered negative grants.
- Data Security Policy: Protects Data Realms by associating them with ACLs.
- Authorization Service: Checks if a privilege is granted to the user in the current session.

Figure 3 illustrates how ACLs can be used to grant privileges to a specific set of protected rows or a Data Realm, for example, my employee record.

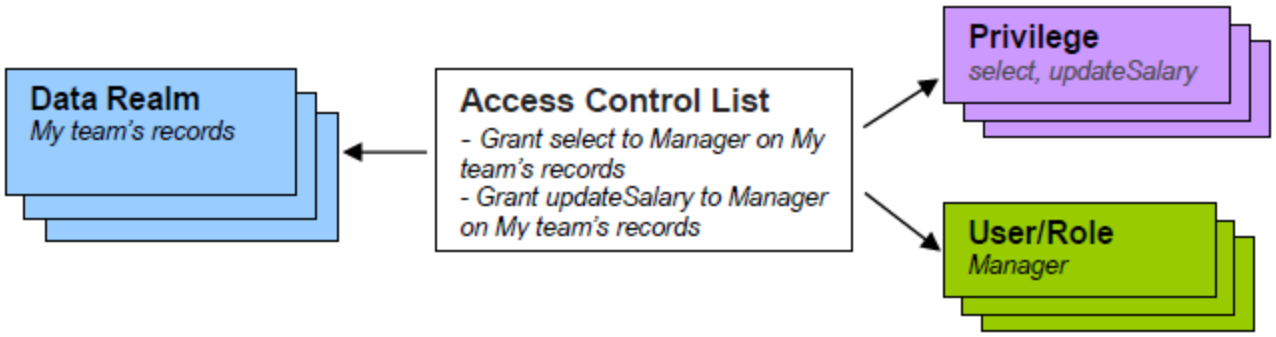

Figure 3. *Components of Oracle RAS data security policy*

Application Users

Oracle RAS introduces the concept of application users within the database to represent application end-users. These users are schema-less and cannot own a database schema or object. However, similar to a database user, an application user can have a default schema for object name resolution. Database privileges can be granted to an application user through application roles.

Application users can be provisioned in an identity store as well as in the database. Application users accessing a 3-tier application are typically authenticated by the middle-tier, and their identity context is security propagated to the database. For 2-tier applications for reporting, patching, maintenance, or other batch programs, application users can also directly connect and be authenticated by the database.

Application Sessions

RAS Application Sessions represent the end users and their security context within the database in a secure and efficient manner. RAS Application Sessions are lightweight as they maintain only the security-relevant state for the user, and have a many-to-one association with a traditional database session. RAS Application Sessions are multiplexed on heavyweight database sessions among many end-users, making it suitable for 3-tier applications with hundreds of thousands of end-users. Oracle RAS introduces the concept of attach and detach of a RAS Application Session to a database session ensuring that only the application user's security context is used for all relevant database operations (see Figure 4).

On first login after authentication, a RAS application session is created for a user encapsulating the user identity and associated roles. Throughout the lifetime of this session, additional identity and authorization context can be associated to the session such as the organization of the user, the application initiating the operation, or the network IP address where the request originated. In a 3-tier web application (see Figure 5), during the course of processing a user request, a database connection is acquired from the pool, and the RAS user application session is attached to the database connection.

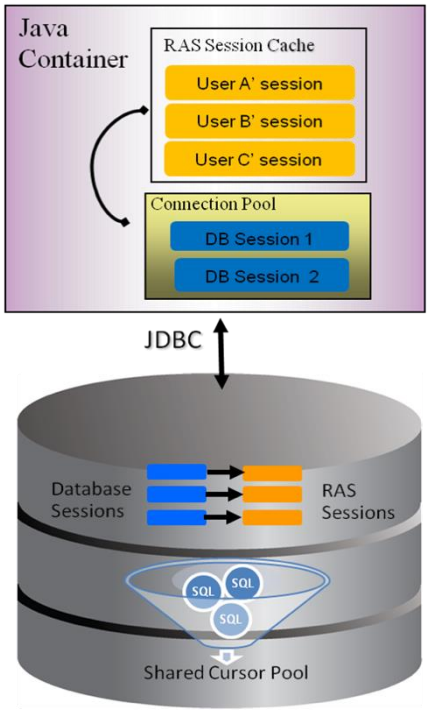

Figure 4. *Attach and detach of RAS applications sessions with database sessions*

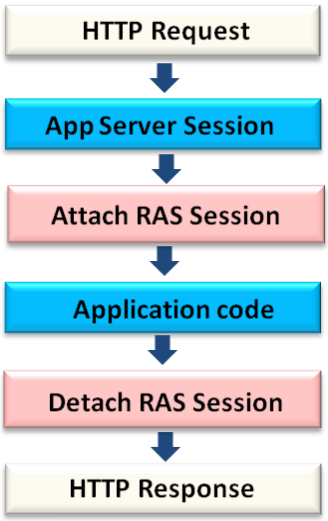

Figure 5. *Attach and detach of RAS session in a web application*

Oracle RAS provides Java application server components, i.e, “Servlet Filter”, which ensures that RAS session is transparently created based on the authenticated user identity. Application framework code attaches the RAS session during the connection acquisition callback of a Java web container. RAS Java components ensure that only the executing user RAS session is attached. RAS sessions are transparently created and cannot be directly accessed by application code. After attach, only the security context in the application session is used to authorize all database operations. RAS sessions use a low privileged database connection primarily as a data channel to the database, and no privileges from the underlying database session associated with the connection pool are used. By constraining all interaction to the database only to the privileges associated with the application user session, RAS essentially enforces the principle of least privilege.

With RAS, Application Session maintains high performance using the following mechanisms:

- **Light-weight Session State:** Application Sessions are not associated with any database resources such as compiled SQL known as server-side cursors (see Figure 4). By keeping only user authorization relevant state and session state in a small memory footprint, RAS is able to support a large number of Application Sessions. As RAS Application Sessions do not own any compiled SQL or cursors, cursors can now be shared across database sessions resulting in additional cursor utilization.
- **Middle-tier Caching:** Application Sessions as well as users’ authorization state are cached in Java middle-tier to improve performance of session attach/detach and authorization-check functionality.
- **Session Context Piggybacking:** RAS session relevant data is piggybacked with application data traffic to the database reducing the overall roundtrips to the database. These roundtrips would be needed if applications were to build its own application session concept in the database to enforce data security.
- **Optimal SQL Plans:** RAS alleviates application developers from the task of SQL tuning for efficient enforcement of application security policies. RAS utilizes various internal techniques to generate optimal SQL statements accessing protected objects.

Application Privileges

RAS Application security policies control application-level operations on application-specific business objects or entities. For example, a Human Resource application can define RequestLeave and ApproveLeave privileges to control the execution of application-level operations and their corresponding SQL actions – select, insert, and update– on Employee Leave records. Oracle RAS provides SQL operators to check if an Application Privilege is authorized for a row of a database table or view.

Similarly, application privileges can be defined to represent cell-level fine-grained object privileges such as ViewSSN that represents SELECT on the SSN column for the executing user’s employee record. Lastly, application privilege can be associated with non database objects such as a UI element for a Task Flow or a Page Flow. In cases when an application privilege does not represent an action on a database object, RAS SQL operators and Java APIs are provided to check the authorization for the application privilege. Thus a RAS application developer can define privileges to also control actions associated with all application-level operations.

Application Roles

Application security policies are typically specified using application roles, without knowing the users that will participate during the execution time of an application. For example, participants of a Manager role can approve leave requests. Oracle RAS can define application roles and grant application privileges to these roles in ACLs. Application roles can be granted to other application or database roles. During user provisioning, application roles are assigned to application users.

Oracle RAS supports a wide range of role-based access control policies as follows:

- **Separation of Duties:** The business can enforce rules such as ensuring that a user cannot both initiate and approve a leave request.
- **Time-based Constraints:** Employees can enroll in stock purchase plan only in November.
- **Delegation Constraints:** A manager role can delegate only certain tasks to a user with AdministrativeAssistant role for a limited duration.
- **State-based Constraints:** RAS supports mechanisms to place constraints on privileges to application users based on their run-time state through roles. For example, an “inside the firewall” role may be enabled for application users connecting from within the corporate firewall, in turn granting them more privileges compared to connecting from outside.
- **Code-based Privilege Elevation:** RAS supports association of privileges to application code through roles. A role could be enabled whenever a sensitive PL/SQL procedure is executed inside the database or a sensitive Java program is run in the middle-tier, in turn allowing an elevation of privileges to complete the sensitive operation.

For example, a payroll user may kickoff a payroll run where the program needs to access employees' payroll data for use in tax calculations, but the user himself may not have authority to access such security sensitive data.

Data Realms

A Data Realm represents a securable business object as a logical collection of data rows in an application table or a view. For example, a business object can be data sets that belong to an organization, department, geography, or related through some other association. The concept of Data Realm captures the typical business scenarios where access control requirements are associated with data sets representing business objects or entities. This collection or dataset is specified using a SQL predicate, where each row of the collection satisfies the predicate. For example, a Data Realm might be all the employee records, records of employees that report to a specific manager, or an employee's own record. In Figure 6, we show these Data Realms for the Employee Detail view. Assuming Nancy is a manager, there are three collections: all the records, records of the employees reporting to Nancy, and her own employee record.

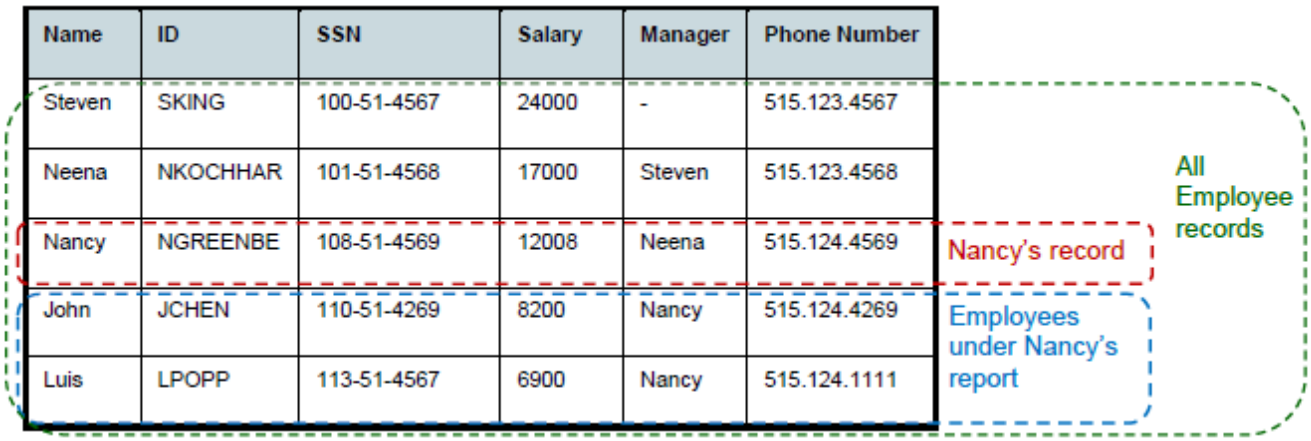

Figure 6. *Sample data realms in employee detail records*

Data Realms are usually represented with SQL predicates and may require joining multiple tables. Based on our security policy modeling experience for various Oracle applications, we have identified the most commonly used data access patterns and provided several types of Data Realms for ease of policy specification as follows:

- **Session Attribute Based Realms:** The rows protected can be selected based on session attributes or application-specific context such as user name in the session or tenant identifier of the application. For the HR application, employees can modify their own contact information. Similarly, a Vice President can view employee salaries of his organization. In these cases, the Data Realm represents a dynamic set of rows depending on the identity context of the executing user. Oracle RAS provides SQL operators for accessing RAS session contexts as well as application defined session contexts.
- **Relational Realms:** Rows of a Data Realm can also be based on join conditions with other tables. For example, a Data Realm can represent all the employees that report either directly or indirectly to the user issuing the query. Here the rows in the Employee table are selected based on the management hierarchy.
- **Master-Detail Realms:** Master-detail is a common data modeling pattern to show a record and its line items. For example, in "Employee Leave" and "Leave Detail" tables, a leave request and its line items in the Detail table may have the same access control policy. Similarly, a row in Employee record and its Job History line items may be protected as a single logical record.
- **Parameterized Realms:** Parameterized Data Realms represent different set of rows based on a parameterized condition in the SQL predicate. For example, sales managers of the east region get access to customer records of the east region, and sales managers of the west region get access to sales records of the west region. Here, the region is parameterized and access to region-specific records is granted to different region-specific managers.
- **Exceptions to Realms:** Once Data Realms are modeled, there can be a set of records that need exception to existing policies. For example, a contract worker may need temporary access to certain employee records.

Access Control Lists

A RAS ACL is a collection of privilege grants or Access Control Entries (ACE), where an ACE grants or denies privileges to a user or a role. By aggregating privilege grants under an ACL, RAS simplifies authorization management. RAS ACLs also allow advanced security policy features such as negative grants, same grants for multiple objects or Data Realms, and various constraints on the grant list such as requiring a user to be in multiple roles to execute a privileged operation.

Data Security Policies

A Data Security policy associates each Data Realm with an ACL. The Data Security policy is thus essentially a collection of Data Realms and their associated ACLs. RAS Data Security policy further supports column-level authorization to mask certain column values based on additional privilege checks. By having both row and column level authorization, RAS Data Security provides cell-level protection.

For column-level authorization, security sensitive columns are associated with RAS application privileges. Once a column is associated with a privilege, the column value for a row can be accessed only if the privilege is granted. In Figure 7, security-sensitive SSN and SALARY columns are first associated with the ViewSSN and ViewSalary privileges. The ACLs are then associated with the Data Realms as shown in Figure 6. In the rows in "Nancy's Reports Data Realm," the manager Nancy can view her employee salaries. For her own employee record Data Realm, Nancy can view her own SSN and salary. Lastly the remaining columns, except SSN and Salary, which are not associated with a privilege, can be viewed by all the employees based on ACL associated with "all Employee" Data Realm. Oracle RAS provides a column indicator on the SQL query result-set indicating if the access to the cell is not authorized. The indicator can be used by the application to mask the column values or display some fixed values with an appropriate format.

Figure 7. *Data realm and column authorization using ACLs*

RAS Authorization Service

Once RAS Data Security is applied on a database table or view, the access control policy is enforced on all SQL statements on all access paths to the object, whether coming in through a 2-tier client-server or any 3-tier application. In addition to support application operation specific authorization on data, RAS provides SQL operators for two additional functionalities:

- The SQL operator ORA_CHECK_ACL executes SQL statements in context of an application privilege. For example, a user can update the leave request records of a table where he has the ApproveLeave privilege.
- The SQL operator ORA_GET_ACLIDS queries the ACLs for each rows of a data security protected table or view. An application developer can use these RAS APIs to determine whether the user is authorized for a specific privilege on a row. For example these APIs can be used to show a button to enable authorized employees to edit their own phone numbers.

The application may also need to specify and enforce a set of functional-level security requirements such as those needed for Task Flows and Page Flows in a Web application. Examples include simple operation checks such as privilege to invoke an item on a menu or clicking an arrow for page navigation. Oracle RAS authorization APIs can be used to make access decisions on such protected resources outside of the database. Without Oracle RAS, such capabilities require custom coding of the access control policy and its enforcement.

ORACLE RAS SECURITY POLICY LIFE-CYCLE

RAS authorization policy is defined during the application development phase, deployed along with the application, and managed during the lifecycle of the application. Figure 8 highlights the tasks that are performed to manage RAS security policies.

At application design time, the architect identifies all the operations that need privileges. Based on application table design and security requirements, tables and views that need Data Security protection are identified, and Data Realms are defined including column protection. The architect then creates a set of application roles and assigns the application privileges to roles in the ACLs used in Data Security policy or functional security.

With RAS, if a connection pool is used by the application to connect to the database, it only needs to use an unprivileged user. With RAS mid-tier Java components, when a user logs into the application, a RAS session corresponding to the user is created and cached in the middle tier. To use RAS sessions, application code needs to call RAS attach/detach session service APIs in connection acquire/release callbacks. RAS session service APIs ensure that only the correct RAS session corresponding to the end user can be attached to the database connection. The application can also use Oracle RAS as an authorization engine to evaluate access rights in the middle tier to display menu items and content.

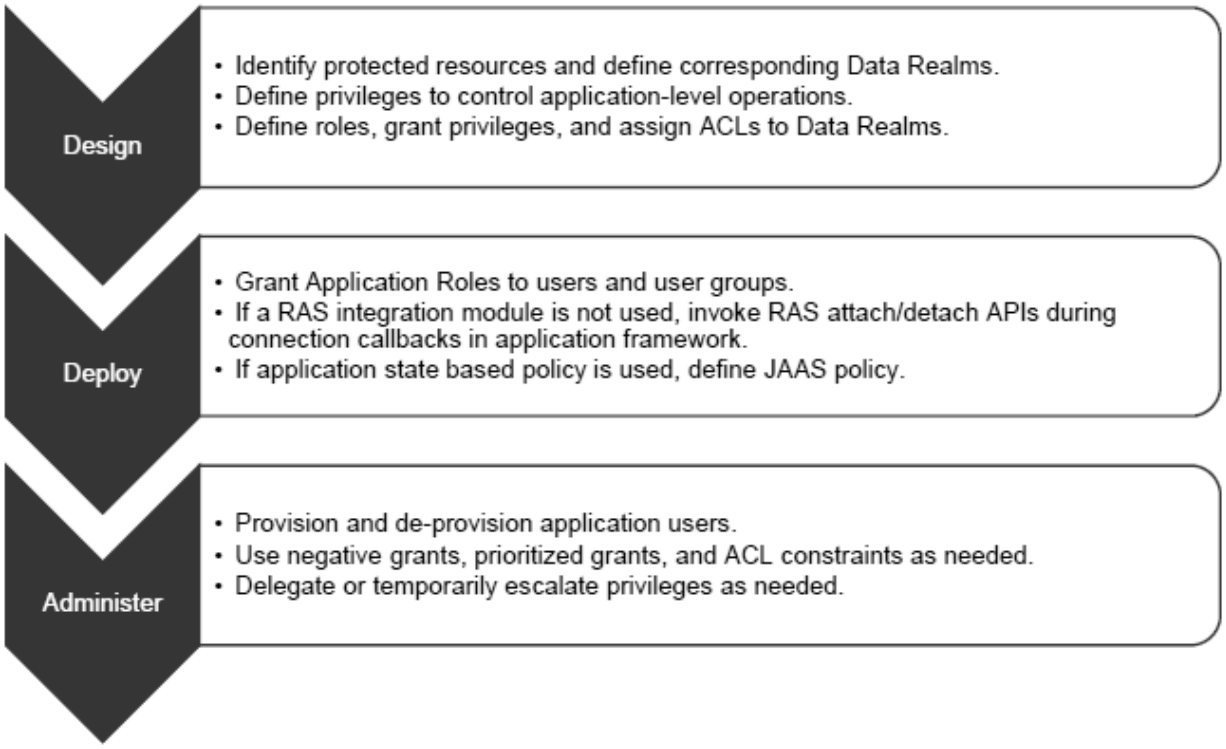

Figure 8. *Stages of Oracle RAS Policy Management*

The application roles exposed by the application cover most use cases, and can typically be mapped to enterprise users and user groups, simplifying the role assignment. In some cases, administrator might identify new roles to be used for additional Data Security or functional security. Administrator may also customize the ACLs to meet specific time-based constraints or provide support for exceptions using negative grants.

EXAMPLE USING ORACLE REAL APPLICATION SECURITY

This example illustrates the Oracle RAS Security framework based on some of the requirements for the HR application discussed at the beginning of this paper. For this sample application, the information about each employee is stored in EMPLOYEES and MANAGERS tables under HRM schema with the following definitions:

```
EMPLOYEES (EMPLOYEE_ID, NAME, SSN, SALARY, PHONE_NO)
MANAGERS (MANAGER_ID, EMPLOYEE_ID)
```

Figure 9. Shows sample employee records without any security policy.

Name	Manager	Phone Number	SSN	Salary
Steven King	-	515.123.4567	100-51-4567	24000
Neena Kochhar	Steven King	515.123.4568	101-51-4568	17000
Nancy Greenberg	Neena Kochhar	515.124.4569	108-51-4569	12008
John Chen	Nancy Greenberg	515.124.4269	110-51-4269	8200
Luis Popp	Nancy Greenberg	515.124.1111	113-51-4567	6900

Figure 9. *Sample employee records in an HR application*

The access control policies we enforce in this example are:

1. Employees can view everyone's name, manager, and phone number.
2. Employees can view only their own Social Security Number (SSN) and salary. They are allowed to update their own phone number.
3. An HR representative can view the SSN of any employee.
4. A manager can view the salary of employees who are directly or indirectly reporting to him/her.

Data Security Using Oracle RAS

Oracle RAS provides PL/SQL administrative APIs and data dictionary views to manage RAS policies. Based on these, we have developed the Oracle RAS Administration tool using Oracle Application Express (APEX) to administer policies. In the remainder of this paper, we use snapshots from the tool to describe the Data Security policy for the sample HR application.

In the above four policy requirements, there are three actors: employee, manager, and HR representative, and three sets of employee records: "records of all employees", "an employee's own record", and "records of employees reporting to a manager". Within these records, the SALARY and SSN columns are security sensitive.

Based on these observations, first we define three application roles: EMPLOYEE, MANAGER, and HRREP. Then we define two privileges: VIEW_SALARY and VIEW_SSN. These privileges are associated with the corresponding SALARY and SSN columns as part of the column authorization in Data Security policy (see Figure 10). We then define the following three Data Realms:

1. ALL_RECORDS: Represented using the SQL predicate "1=1", which is true for all rows of the table.
2. MY_RECORD: The record of the executing user is identified based on the following predicate using Oracle RAS session context check operator. As shown below, the operator returns the session's logon user identifier.
3. EMPLOYEE_ID= XS_SYS_CONTEXT('XS$SESSION', 'USERNAME')
4. MY_REPORTS: All employees reporting to a manager are found based on a hierarchical CONNECT BY query on the Manager table.

Appropriate privileges on these Data Realms are granted to the roles through ACLs as shown on the right side of Figure 10. On all employee records, the Employee role is granted the SELECT privilege and the HRREP role is granted the VIEW_SSN privilege. As SSN and SALARY columns are associated with privileges, the SELECT privilege only grants access to remaining columns of the Employee table. Similarly, on employee's own record, the Employee role is granted the VIEW_SSN, VIEW_SALARY, and UPDATE privileges so that an employee can view security sensitive columns of his/her own record as well as can update the phone number. Lastly, the Manager role is granted the VIEW_SALARY privilege on the SALARY values of his/her employees.

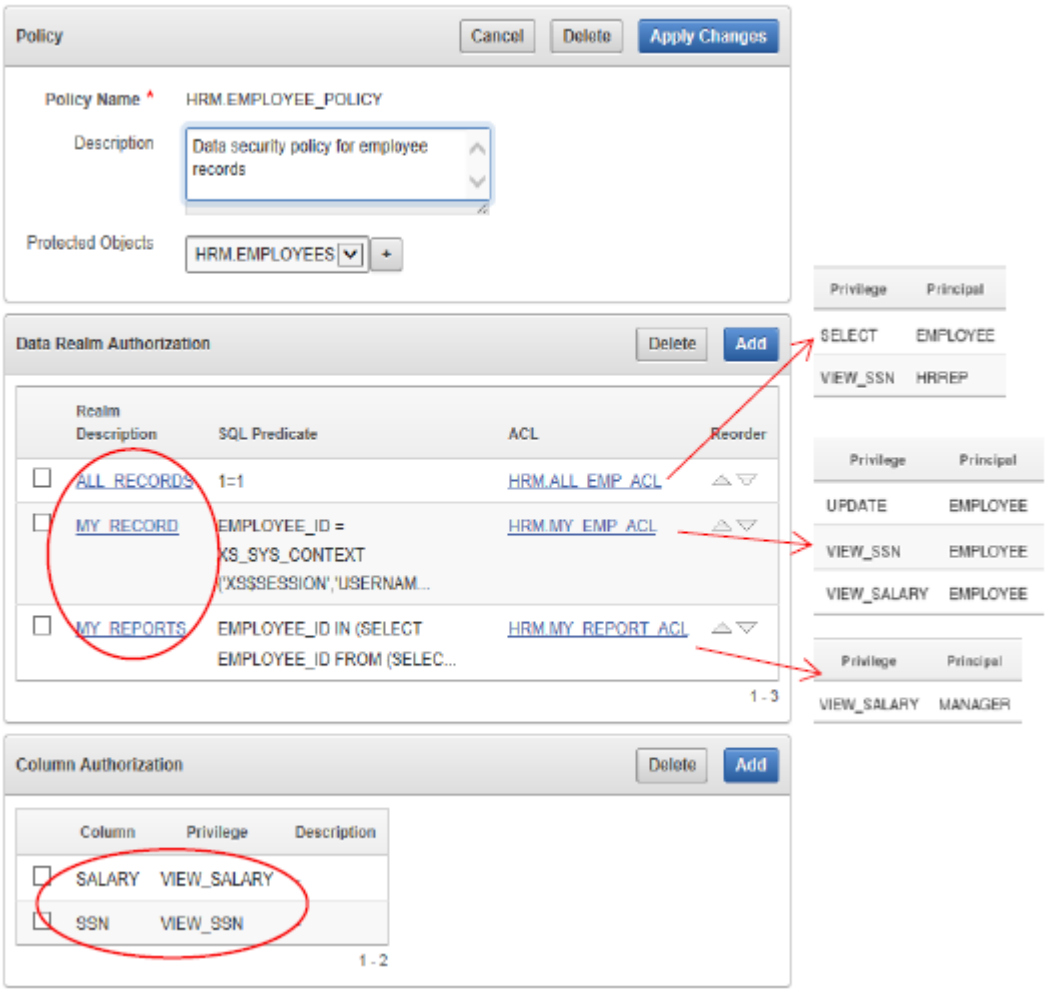

Figure 10. *Data security policy for employees table*

Figure 11 shows the role centric view of the privileges granted on the Data Realms in Figure 10. These grants represent the four authorization requirements for the sample HR application.

Role	Description	Privilege	Object	Data Realm
EMPLOYEE	Regular employees	SELECT	HRM.EMPLOYEES	ALL_RECORDS
EMPLOYEE	Regular employees	UPDATE	HRM.EMPLOYEES	MY_RECORD
EMPLOYEE	Regular employees	VIEW_SALARY	HRM.EMPLOYEES	MY_RECORD
EMPLOYEE	Regular employees	VIEW_SSN	HRM.EMPLOYEES	MY_RECORD
HRREP	HR representatives	VIEW_SSN	HRM.EMPLOYEES	ALL_RECORDS
MANAGER	Managers	VIEW_SALARY	HRM.EMPLOYEES	MY_REPORTS

Figure 11. *Roles and privilege grants on data realms*

During the provisioning time of this application, we assign the Employee and Manager roles to Nancy. Figure 12 shows the values of the cells that Nancy can view. In this report, Nancy can view all columns of her own record, salaries of her employees (John and Luis in Figure 11), and public information of all other employees. However, she cannot view SSN of other employees and salaries of employees who do not report to her. The unauthorized SSN cell values are masked with "111-11-1111" and unauthorized SALARY cell values are masked with "xxxxxx" using RAS column masking operator.

Name	Manager	Phone Number	SSN	Salary
Steven King	-	515.123.4567	111-11-1111	xxxxxx
Neena Kochhar	Steven King	515.123.4568	111-11-1111	xxxxxx
Nancy Greenberg	Neena Kochhar	515.124.4569	108-51-4569	12008
John Chen	Nancy Greenberg	515.124.4269	111-11-1111	8200
Luis Popp	Nancy Greenberg	515.124.1111	111-11-1111	6900

Figure 12. *Nancy's query results on employee table with RAS data security.*

INTEGRATION WITH APPLICATION DEVELOPMENT PLATFORMS

To reduce the development effort to uptake Oracle RAS, RAS runtime session is integrated with Oracle Application Express (APEX 5.0) and Oracle Platform Security Service (OPSS) distributed with Fusion Middleware 12.1.3. In the following, we discuss these two out-of-the-box RAS session integration modules. Same strategy can be used for application platform specific development of RAS session integration using RAS java and PL/SQL APIs as RAS recognizes an end-user defined in an external identity store.

Native Integration with Java EE

Oracle RAS is integrated with Java Enterprise Edition (Java EE) web applications that are supported by OPSS (Oracle Platform Security Service) as its application security provider. In this integration, Oracle RAS is deployed in Java EE web applications along with OPSS. Based on the web user's container authentication context, OPSS computes an application security context with the user's application roles and authorization attributes. This context is then extended and used by Oracle RAS for all authorization decisions. Figure 13 shows the runtime deployment of RAS with OPSS.

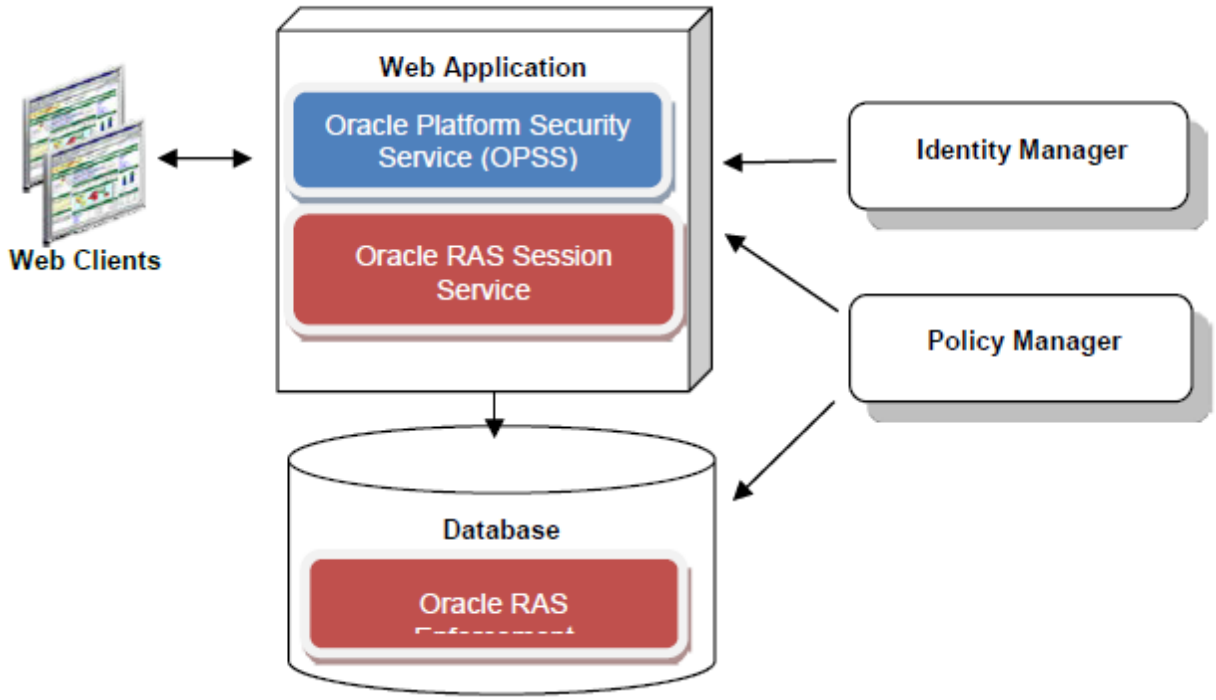

Figure 13. *Web application deployment with RAS and JAVA EE.*

Native Integration with Oracle APEX

Oracle Application Express (APEX) is a web browser based development tool for rapidly building declarative, database-centric, 2-tier web applications. Oracle APEX does not directly provide any data access control functionality and expects APEX developers to implement access control artifacts such as application privileges, roles, and fine grained access control on data or application defined business objects. Oracle RAS is integrated with Oracle APEX 5.0 providing extended access control features for APEX applications. Here RAS session is created and managed transparently and natively within APEX application development framework (see Figure 14). With this framework, APEX application code in the database runs within RAS session context. To enforce access control on data, APEX application developers do not need to write any application runtime code for RAS policy enforcement. To render UI items based on RAS access control policy, RAS privilege check operators can be used in APEX declarative authorization rules.

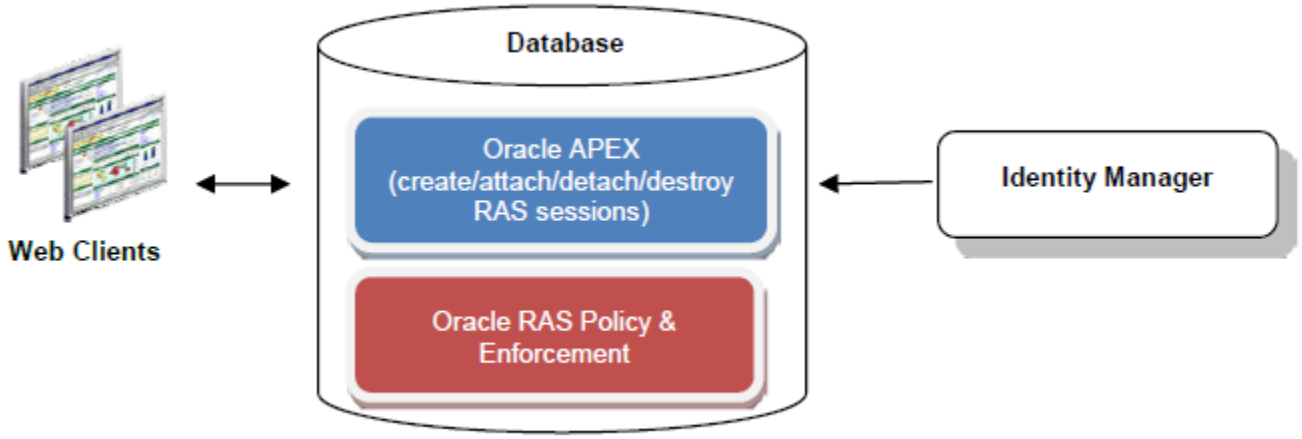

Figure14: APEX Applications with Oracle RAS

Figure 14. *APEX applications with Oracle RAS.*

ORACLE RAS EXTENDED FEATURES

Oracle RAS can enforce a wide range of access control requirements for applications as follows:

Application Requirement	Security Feature Needed	RAS SUPPORT
A Vice President wants to delegate salary administration to his assistant.	Delegation support for specific tasks	Application roles support delegation and time-based limits.
A contract worker needs temporary access to certain employee records.	Exception to existing policies	RAS supports per-row ACL for each row of a table. Adding the higher-level policy as per-row ACL overrides existing policies.
Rights granted to a project owner should be revoked at the end of the project.	User-specific time-based privilege revocation	RAS supports time-based constraint on users. A user can have time-based roles and grants.
Employee can enroll in stock purchase plan only within the specified enrollment period.	Role-specific time-based authorization	RAS supports time-based constraints on roles. A stock purchase plan role can be granted to all employees and enabled only within the specified time period.
Manager should not be able to both initiate and approve promotions.	Seperation of duties	RAS roles in conjunction with realm predicate can express a wide range of role-based constraints
Only a Vice President with Donor previliges can approve the corporate donations.	Role-coupling constraints	RAS ACLs can be constrained with additional ACLs enforcing that the same privilege must be granted to multiple roles for an operation.

Sensitive salary data can only be viewed from inside Corporate firewall	State-based authorization	State-based roles can be used to enforce additional access controls.
Compensation Portal can only be accessed by Managers.	Operation protected with application-defined privileges	RAS supports access control on such application-specific operations using ACLs.
A new regulation requires treating home address as sensitive.	Extensible access control framework for both rows and columns	RAS supports declaration of new privileges and modification of ACLs and other access control policy components.
HR and ERP applications need to enforce a common policy when the user is outside the corporate firewall, or when the user is not strongly authenticated.	Common authorization policy	RAS supports system-wide ACLs that can grant or deny privileges for all the deployed applications.
Auditing end-users activities.	Auditing application users' activities	Application Session is natively integrated within database resulting in auditing end user details in database audit logs.

ORACLE REAL APPLICATION SECURITY (RAS) BENEFITS

Oracle Database 18c provides the next generation authorization architecture for applications through Real Application Security:

- **Secure End-User Identity Propagation:** Application sessions allow the end-user identity and associated attributes to be conveyed securely to the database allowing the database to use the information for end-user access control and auditing.
- **Uniform Data Security:** The RAS Security model allows uniform specification and enforcement of access control policies on business objects irrespective of the access path. It overcomes the limitation of custom-built approaches that only work when an object is accessed via the specific code path that has access control logic embedded into it.
- **Declarative and Fine Grained Access Control:** RAS policy components encapsulate the access control requirements of the application in the form of declarative policy on data for application users, application roles, and application privileges. With column security, the RAS model extends authorization to the column level to protect sensitive data such as SSN. With support for master-detail, parameterized, delegation, and exception-based declarative policies, RAS meets the real-life deployment requirements of applications.
- **Security without Performance Trade-off:** In most current systems, security is either coded into the applications or it is externalized but requires multiple round-trips impacting performance. Unlike these cases, RAS is natively implemented in the database and provides a security solution without trading-off performance.

While the paper focuses on 3-tier applications, Oracle RAS secures all applications that access data, including stand-alone client-server applications. The applications do not have to develop their own access control policy infrastructure within the database. Administration of access control policy is now separate from program code, and becomes flexible and extensible.

Finally, RAS unifies database and application-specific access control models by making it possible to define and use application-specific privileges, users, and roles within the database. RAS provides the long-needed application authorization functionality in the database and a uniform administration model for access control policies on data. In Figure 15, benefits for using Oracle RAS for different users who are concerned with securing applications are presented.

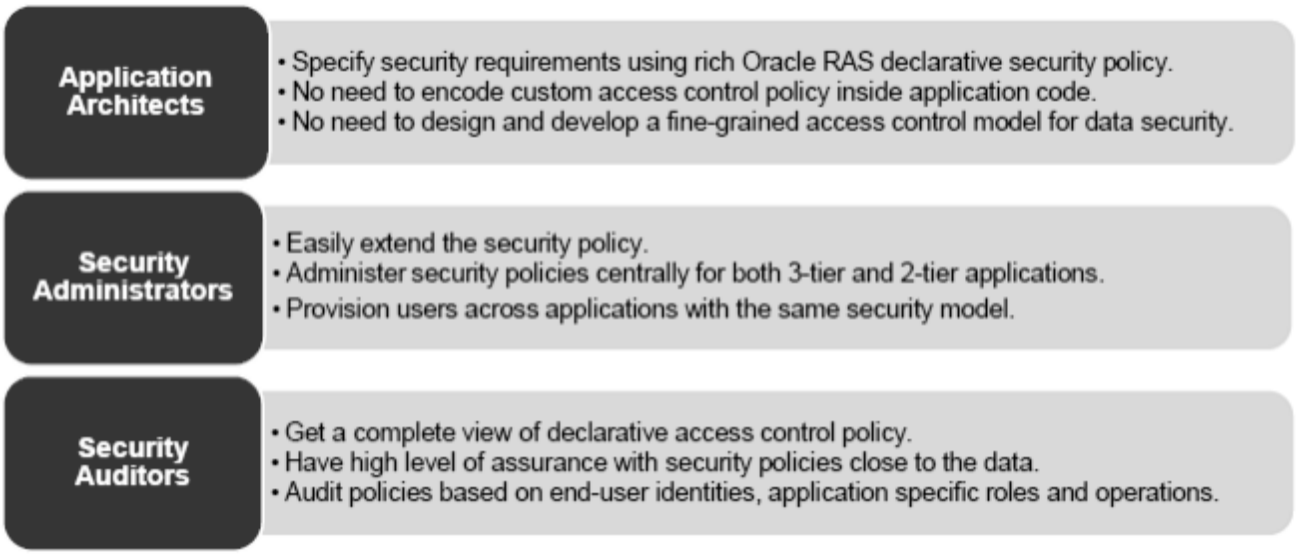

Figure 15. *Oracle RAS security benefits for different types pf users.*

RAS has benefited greatly from Oracle's experience of building packaged applications, and is designed to support wide-variety of authorization use cases for different modules while keeping security, scalability, performance, and manageability into consideration

WHAT NEXT

This document discusses the motivation and concepts behind Oracle Real Application Security. The Oracle RAS development framework includes PL/SQL APIs for policy administration and Java and PL/SQL APIs to enforce RAS security policies for 3-tier and 2-tier applications. Please consult the Oracle Real Application Security Architecture paper and Developer Guide for further information.