Group 15

CS5322 Project 2 Report

Members:

Name	Matric
Chun Hong Wei	A0167886E
Chew Zheng Xiong	A0167003R
Ng Jong Ray, Edward	A0216695U
Cheng Xianhao	A0167000X
Li YingHui	A0112832R

Contribution statement: Everyone had an equal and fair contribution

Introduction

To thrive in a competitive market, modern businesses often turn to Enterprise Resource Planning Systems (ERP) as indispensable tools for efficient and comprehensive management. These systems automate and streamline processes in big organisations which often involve multiple parties by integrating the various functions of an organisation through an unified system. While the benefits of such integrations are often talked about, it is also important to address new challenges in the area data privacy – such system has to ensure that restrictions are finely tuned so that employees have just the right amount of access to perform their tasks, but not too rigid such that granting additional access is made impossible or unreasonably difficult. Moreover, it is paramount that companies adhere to secure data security principles so as to protect company information and secrets from unauthorised access and manipulation.

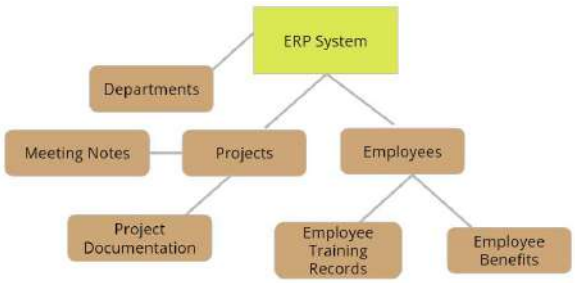

Figure 1.1: ERP Prototype System Functionality Illustration

Taking into account such nuanced data access requirements in ERP systems, the team has decided to recreate a simple ERP system database (as depicted in Fig 1) with Oracle database which comes with a multilevel security feature out of the box Oracle Label Security (OLS). The aim of the project is to demonstrate how multilevel security through OLS can be used to help secure company data by providing granular yet flexible access control in settings where data is shared across multiple users physically but access to part of the data should be restricted logically to a set of users.

Application

This ERP system database prototype is designed to serve company employees, mainly Employees from the HR and IT departments. The ERP is capable of storing and retrieving employee details, including their training records and employee benefits, department information, company projects and as well as meeting notes. The ERP system aims to serve employees and managers alike in a multitude of cases.

To simulate company onboarding, a new user will be tagged with their respective department and security level when receiving their account.

Employees who use the ERP system, be it from HR or IT departments, should be able to use the system to get and update information about their details as well as let the employee know what company benefits they are entitled to. Moreover, the ERP system aims to help them track their professional progression by giving them more details of their completed and pending training programs. Employees should also be able to view related meeting notes.

Our ERP system aims to allow managers of the departments to also streamline their work by managing their team and projects more effectively. This can come in the form of viewing information of their team such as their training records and details, and also manage morale with the availability of employee benefits to each tier of employees. Moreover, managers can better track the progress of their projects through the ERP system as well.

OLS Implementation

The OLS implemented in our ERP system database is anchored by a policy container that ensures read and write controls are enforced on the 'general_ols_col' column, which is a part of each table within the ERP system database when the OLS policy created is applied to the table.

Firstly, our ERP system defines three principal security labels to classify the sensitivity of our data objects. The lowest sensitivity label is 'RESTRICTED' with a level number of 100, 'CONFIDENTIAL' at level 200, and the highest sensitivity is 'SECRET' at level 300. The security levels assigned help to prevent “read up” as users with a lower security level such as 'RESTRICTED' would not be able to read data tagged with a higher security level such as 'CONFIDENTIAL' or 'SECRET', as seen in Figure 3.1.

Figure 3.1: Security Levels

Next, to allow for more refined data access within each security label, a range of compartments were introduced. This helps to prevent employees with higher security levels from viewing data that is not related to their role. The compartments correspond with the various data domains within the organisation. This includes: 'MEDICAL', 'FINANCIAL', 'RETIREMENT', 'TECHNICAL', 'MANAGERIAL', 'SAFETY', 'REPORT', 'BLUEPRINT', 'BUDGET', 'STRATEGY', and 'REVIEW'. Each of the compartments are assigned unique numbers from 100 to 304, enabling granular control over the different sensitive information.

Moreover, our ERP system can help enforce organisational roles and functions with regards to data access. This was done by creating OLS groups which allow for a hierarchical structure.

One OLS group implemented in the ERP system database was created to reflect departments in the organisation. There are two main departments in this organization - Human Resource (HR) and Technology (IT). IT is split to one parent and one child department, which are IT Global (Parent) and IT Support (Child).

The department labels start with COMPANY, which represents the executives in an organisation which have access to data belonging to all departments. This is followed by the management level of two main departments 'HUMAN_RESOURCE_MANAGEMENT' and 'TECHNOLOGY_GLOBAL_MANAGEMENT' to represent the department managers role. Next is either the employees roles of each division themselves, as seen in the case of HR where the following label is HUMAN_RESOURCE_RECRUITERS, or sub-divisions within a department as seen in IT where the following labels are TECHNOLOGY_SUPPORT_MANAGEMENT and TECHNOLOGY_GLOBAL_ENGINEERS to represent the middle managers in each sub-division. In IT, this is finally followed by the employee groups TECHNOLOGY_SUPPORT_TECHNICIAN and TECHNOLOGY_GLOBAL_TECHNICIAN respectively.

Each of the groups is identified by their own unique group numbers and some groups are affiliated through a parent-child relationship. This label-based access is able to reflect the departmental hierarchy and organisational hierarchy to better manage user access to the ERP, as seen in Figure 3.2 which shows the group hierarchy for parent-child relationship.

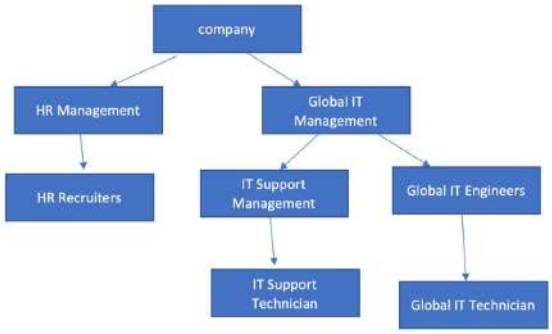

Figure 3.2 Group hierarchy

Lastly, the OLS policy also extends to the individual database user accounts and not just the data objects. Minimum and maximum security levels based on the job roles have been efficiently assigned for all users to ensure that their access to data is controlled. A default session level has also been assigned for each user as well.

For example, user 'globalit_1' is assigned a maximum security level of 'SECRET' and is associated with the 'TECHNOLOGY_GLOBAL_MANAGEMENT' group, while 'hr_1' also retains a 'SECRET' level but is linked to 'HUMAN_RESOURCE_MANAGEMENT'. This shows the approach to role-based access control based on the granularity of the users.

Moreover, we have added 'label_default' under the default_options parameter of the policy. When a user inserts data into any table, the data object will be tagged with the same OLS labels as what the user will have in the current session. This is to help prevent a “write down” situation where a user with 'SECRET' level privileges inserts data with only a 'RESTRICTED' label.

The SQL code for the above OLS setup can be found in Appendix A of this report.

Overall, the OLS implementation in our ERP system database focuses on preventing illegal information flow.

Security Policies

Other than the implementation of OLS, other access control strategies have been implemented to support access control of the ERP system. These additional access control measures include Discretionary Access Control (DAC) and Virtual Private Database (VPD). The DAC implementation is structured around the two primary user roles within the ERP system, which are namely employee and manager. The employee roles include: HUMAN_RESOURCE_RECRUITERS, TECHNOLOGY_GLOBAL_ENGINEERS, TECHNOLOGY_SUPPORT_TECHNICIAN or TECHNOLOGY_GLOBAL_TECHNICIAN. These roles that are defined determine the level of access to each of the database tables. VPD on the other hand is implemented for certain tables within the ERP system where users should only be able to access their own data.

Discretionary Access Control

Firstly, an employee has been granted 'SELECT' privileges on tables such as PROJECTS, PROJECTDOCUMENTATION, MEETINGNOTES, EMPLOYEES, EMPLOYEEBENEFITS and EMPLOYEETRAININGRECORDS. By granting the employees’ access rights in this manner allows relevant users assigned the employee role to view and access data across a range of operations within the ERP. This facilitates sufficient privileges for employees to obtain the necessary information to perform their job functions.

In addition to read permissions, the employee role is also granted 'INSERT' and 'UPDATE' rights on some of the tables mentioned previously. The limited write access is also granted to enable employees to contribute to the relevant tables of documentation and maintain current records within their job scope.

In order to provide more privileged access, the DAC is specially designated to the manager (parent) roles. This also includes all the 'SELECT' privileges of the employee (child) roles. This provision of modified access to this table for managers allows them to view training records of their subordinates, which is essential for overseeing their development and performance within the organisation. Furthermore, users with similar levels to others can also view the training records of their peers, to increase awareness. This is further explained in the Database section below.

Furthermore, the manager role has also been granted 'INSERT' and 'UPDATE' permissions on all tables. This additional DAC takes into account the performance of managerial duties

which involve updating project statuses, providing edits to meeting documentation and the management of employee records.

A summary of the DAC policies can be found in table 1 below.

To further illustrate the various role-based permissions in practice, organisational user accounts have been created, as illustrated in the table below.

Account	**Roles**
ITSUPPORT_1	EMPLOYEE
GLOBALIT_1	MANAGER
HR_1	MANAGER
HR_2	EMPLOYEE
GLOBALIT_2	EMPLOYEE (Engineer)
ITSUPPORT_2	MANAGER
GLOBALIT_3	EMPLOYEE (Technician)

One user to highlight would be 'ITSUPPORT_1' who has been assigned an employee role. This will showcase the read and limited write permissions for the ERP tables. Similarly, 'GLOBALIT_1', 'HR_1' and ITSUPPORT_2 have been created and assigned a manager role. The hierarchical roles reflect the requirement for broader data access within ERP databases, which also includes the Engineers and Technicians.

The DAC implementation has been carried out to ensure that it aligns with the principles of least privilege and separation of organisational duties This ensures that users of the ERP system are only given the level of access that is necessary for their job functions. Overall, DAC implementation in our ERP system helps to quickly govern if users are allowed to access data based on their role and table which they are accessing data from.

Table	**Database Roles**	
	MANAGER	*EMPLOYEE*
projects	S, I, U	S
project_documentation	S, I, U	S, I, U
meeting_notes	S, I, U	S, I, U
employee_training_records	S, I, U	S
employee_benefits	S, I, U	S
employee	S, I, U	S, I, U
S = SELECT, U = UPDATE, I = INSERT		

Table 1: DAC policies applied to each table

Virtual Private Database

The ERP system that is implemented with OLS enforces fine-grained access control, ensuring data security and compliance with the organisational security policies. VPD has also been implemented to complement the OLS by dynamically appending predicates to SQL statements based on the security policies which are linked to the user properties. These user properties include the 1) **username** which are identified in both the EMPLOYEES table and the database account, retrievable via USERENV namespace using the SESSION_USER parameter 2) the user's **position** stored in the POSITION column in the EMPLOYEES table and accessible through a custom context. The position property is automatically loaded into the context using database logon triggers.

The layer approach to security incorporating OLS, VPD and DAC implementation ensures that the ERP database adheres to the principle of least privilege and separation of job-related duties minimising the risk of illegal information flow.

Database Tables

With the OLS labels described above implemented in our ERP system, this section will be devoted to showcasing how the multitude of security access controls will affect the key tables in our ERP system database. In all of our tables, the security level, compartments and groups have their own column for the ease of explanation. In a realistic deployment, not all the information will be shown.

The EMPLOYEES table stores personal information about the employees and is associated with each employee. This includes information such as their names, contact information, security level, position and department. The position and department columns denote the employee's role and organisational group. This is aligned with the implemented OLS which can be used to control access to this table based on the security level, position and department of the user. Since all tuples are coded with 'RESTRICTED' label, this would mean that any user will be able to view all employee information. This would also not be solved if we changed the security levels of the employee data as a 'SECRET' level user would then be able to view all employee data. Hence, a VPD implementation (in Appendix C) is required for more fine-grained access control. Our VPD ensures that normal IT employees will only be able to view their own employee records while IT managers would be only able to view only the information of employees in their department. Meanwhile, HR employees would still be able to view all the employee data for their job. Utilising a VPD here allows us to provide more intricate access control on the EMPLOYEES table, giving different types of users different types of accesses based on their job requirements.

Next, the DEPARTMENTS table stores information regarding the different departments that are part of the company.

Following which, the PROJECTS table stores information about the project ids, project names, start date and end date to help determine which projects are ongoing and which projects that have been undertaken by the company have been completed. The OLS can be used to determine user access based on their department and security level of the project. OLS labels used in this table primarily revolve around security labels and group labels to represent the sensitivity of each project and the department undertaking the projects. Combined with our implemented DAC, employees in each department should not be able to view projects that are undertaken by other departments, and that employees can only view projects that are equal or lower than their security clearance in the organisation. Managers on the other hand are able to view, add and update projects which are related to their department and for their own security level as well.

Moreover, the PROJECTDOCUMENTATION table stores the documentation of the projects that have been undertaken by the company. This closely follows the PROJECTS table by including a foreign key relation to the PROJECTID column in the PROJECTS table, ensuring that a project document must be tied to an existing project. Here a project document will contain its security label, department group, as well as the compartments which this document falls into. Here, some compartment categories are its document type such as 'FINANCIAL', 'TECHNICAL' and 'STRATEGY'. Another compartment is determined by the datatype of the document as well such as 'REPORT', 'BLUEPRINT' and 'BUDGET'. Along with the security label and department, the OLS label will determine the user access to this table.

The EMPLOYEEBENEFITS table stores information regarding the generic employee benefits that are made available to the employees throughout the company, depending on their department, employee level (based on their security access), benefit type, benefit amount and benefit form. The OLS enables the HR department to have wider access in order to make changes to the benefits available, based on their individual groups and compartments as defined.

The MEETINGNOTES table stores information of meetings that have been held in the company. Here a meeting will contain its security label, department group, as well as the meeting type compartment. The compartment of either 'REVIEW' or 'STRATEGY' enables more fine-grained access control to ensure that users are able to only see meeting notes which correspond to their security label, department hierarchy and compartment as well.

Lastly, for the EMPLOYEETRAININGRECORDS table, it stores information regarding the type of training and the training certification(s) obtained by an employee. An employee is able to view their own training record if it is within their security clearance level. This is intended as there might be training which the employee might have been given and passed which they should not know about. Because our detailed and robust OLS group combines both departments and organisation hierarchy, we are able to enable flexible access controls to fit our ERP system requirements. A user from IT Global can view training records of another user from IT Support while any user from the HR department would be able to view the records of users from other departments.
The inbuilt organisational hierarchy allows managers to view employee training records from their department while employees are unable to view the training records of managers. Moreover, since the 'ENGINEER' role is a parent to the 'TECHNICIAN' role, this allows an Engineer to view the training records of other Engineers and Technicians as well. Importantly, we intentionally allowed users who have similar roles and security labels to view each other's training records as a form of motivation.

Additionally, the OLS compartments implemented for Employee Training Records are 'TECHNICAL', 'MANAGERIAL' and 'SAFETY'. Similarly, this helps with implementing fine-grained access control. Only an engineer with a 'SAFETY' label would then be able to view records containing the 'SAFETY' compartment.

Overall, utilising OLS labels as well as other access control policies differently for the different tables allow for varied use cases which our ERP system database requires.

Results & Discussion

Firstly, the application of OLS has effectively prevented unauthorised access and any potential security breaches to the ERP system. This reinforces the robustness of the system in protecting sensitive organisation data being exchanged at a high rate throughout the work day.

However, we understand that further improvements to the table design is required to make this a fully usable ERP solution for small enterprises, as it lacks several tables such as matching each employee to their entitled employee benefits, for example. This is not included as it is beyond the scope of this project to demonstrate the effectiveness of OLS policies on the discretionary access controls.

Performance wise, the implementation of the OLS did not impede the efficiency of the ERP system. Despite the variations in query execution times, the differences were not substantial in illustrating the degradation of performance over time. This further highlights the capability of OLS in balancing between providing a high level of security and operational efficiency.

Next, there is a need to consider the scalability of the ERP system as the organisation continues to grow. The ERP system will become increasingly complex with more tables being incorporated, which requires the system continuously adapt. There will be the need to modify the OLS implementation in order to address the evolving organisational structure and new security risks.

Also, the integration of OLS has enabled organisational security to be achieved. With the application of OLS, detailed security labels and

role-based access control has been crucial in managing user access and the exchange of information across the organisation. This has been implemented in accordance with the principles and least privilege and separation of duties. The OLS approach adopted minimises the risks of illegal information flow by preventing users of a lower security level from reading data from a higher security level. Moreover, our OLS implementation can also prevent users of high security level to bypass access control by reading, for example, a 'SECRET' document then "writing down" as data with a lower security level. This can be done to ensure the integrity and confidentiality of the data.

While the currently implemented OLS, DAC and VPD provide robust access control, the ERP could potentially integrate Dynamic Data Masking (DDM) to further enhance data privacy. DDM is able to dynamically obscure specific data within a database. This ensures that the sensitive data is not exposed to unauthorised users when they access the database. This is particularly useful for scenarios where employees need to access databases containing sensitive information but visibility of all data is not required. For example, employees in HR are required to access personal details of staff but are not required to see their salary information. The potential implementation of DDM can mask these sensitive details dynamically as determined by the user roles and access levels within the ERP.

Lastly, the multiple implementation of OLS, DAC and VPD, adds a layer of complexity in managing security policies for the administrator. The implementation of multiple security policies requires rigorous planning and regular updates to ensure policies remain relevant. Especially for organisations with ever evolving structure and data access needs based on its growth. Furthermore, the additional layer of complexity will necessitate extensive training for database administrators and end users to better understand how to best utilise the ERP system. This could be resource-intensive, which will be incurred by the company.

Conclusion

This project has demonstrated the effective implementation of OLS in an ERP system, which has been tailored to meet the various data access and privacy needs. The ERP system encompasses multiple departments and different user roles to illustrate a robust framework for managing sensitive data through a multi-layered security approach. By integrating OLS, DAC, and VPD, the established ERP system adheres to stringent data security standards and offers flexibility and scalability required by the company.

Next, the implementation of tiered security labels, refined data access compartments, and role-based access control have also effectively addressed the complex requirements of data confidentiality and integrity. This enables the ERP system to share sensitive data securely across different organisational levels and users. The implementation also highlights the importance of a balanced approach to security and operational efficiency, ensuring that enhanced data protection does not impede system performance.

In conclusion, this project highlights the capability of OLS in ensuring the data security within ERP systems. This project potentially serves as an example for organisations to develop a scalable model aimed at protecting sensitive data while maintaining high operational efficiency. This project also illustrates the contributions to data security in ERP systems, offering a practical and effective solution to modern-day security challenges.

Appendix A: OLS Setup

```
-- Create policy container
BEGIN
    SA_SYSDBA.CREATE_POLICY (
            policy_name      => 'general_ols_policy',
            column_name      => 'general_ols_col',
            default_options  => 'read_control, write_control,
label_default');
END;

-- Create security policy labels
BEGIN
    SA_COMPONENTS.CREATE_LEVEL (
            policy_name => 'general_ols_policy',
            level_num   => 100,
            short_name  => 'R',
            long_name   => 'RESTRICTED');

    SA_COMPONENTS.CREATE_LEVEL (
            policy_name => 'general_ols_policy',
            level_num   => 200,
            short_name  => 'C',
            long_name   => 'CONFIDENTIAL');
    SA_COMPONENTS.CREATE_LEVEL (
            policy_name => 'general_ols_policy',
            level_num   => 300,
            short_name  => 'S',
            long_name   => 'SECRET');
END;

-- Create compartments
BEGIN
    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'MEDICAL',
            short_name       => 'MED',
            comp_num         =>  100);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
```

```
            long_name        => 'FINANCIAL',
            short_name       => 'FIN',
            comp_num         =>  101);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'RETIREMENT',
            short_name       => 'RET',
            comp_num         =>  102);
END;

BEGIN
    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'TECHNICAL',
            short_name       => 'TEC',
            comp_num         =>  200);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'MANAGERIAL',
            short_name       => 'MAN',
            comp_num         =>  201);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'SAFETY',
            short_name       => 'SAF',
            comp_num         =>  202);
END;

BEGIN
    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'REPORT',
            short_name       => 'REP',
            comp_num         =>  300);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'BLUEPRINT',
            short_name       => 'BLU',
            comp_num         =>  301);
```

```
    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'BUDGET',
            short_name       => 'BUD',
            comp_num         =>  302);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'STRATEGY',
            short_name       => 'STR',
            comp_num         =>  303);

    SA_COMPONENTS.CREATE_COMPARTMENT (
            policy_name      => 'general_ols_policy',
            long_name        => 'REVIEW',
            short_name       => 'REV',
            comp_num         =>  304);
END;

-- Create Groups
BEGIN
    SA_COMPONENTS.CREATE_GROUP (
            policy_name     => 'general_ols_policy',
            group_num       => 4000,
            short_name      => 'COMPANY',
            long_name       => 'COMPANY');

    SA_COMPONENTS.CREATE_GROUP (
            policy_name     => 'general_ols_policy',
            group_num       => 4100,
            short_name      => 'HR_MGR',
            long_name       => 'HUMAN_RESOURCE_MANAGEMENT',
            parent_name     => 'COMPANY'
            );

    SA_COMPONENTS.CREATE_GROUP (
            policy_name     => 'general_ols_policy',
            group_num       => 4110,
            short_name      => 'HR_REC',
            long_name       => 'HUMAN_RESOURCE_RECRUITERS',
            parent_name     => 'HR_MGR'
        );
```

```
SA_COMPONENTS.CREATE_GROUP (
        policy_name     => 'general_ols_policy',
        group_num       => 4200,
        short_name      => 'IT_GLOBAL_MGR',
        long_name       => 'TECHNOLOGY_GLOBAL_MANAGEMENT',
        parent_name     => 'COMPANY'
    );


SA_COMPONENTS.CREATE_GROUP (
        policy_name     => 'general_ols_policy',
        group_num       => 4210,
        short_name      => 'IT_SUPPORT_MGR',
        long_name       => 'TECHNOLOGY_SUPPORT_MANAGEMENT',
        parent_name     => 'IT_GLOBAL_MGR'
    );

SA_COMPONENTS.CREATE_GROUP (
        policy_name     => 'general_ols_policy',
        group_num       => 4211,
        short_name      => 'IT_SUPPORT_TEC',
        long_name       => 'TECHNOLOGY_SUPPORT_TECHNICIANS',
        parent_name     => 'IT_SUPPORT_MGR'
    );

SA_COMPONENTS.CREATE_GROUP (
        policy_name     => 'general_ols_policy',
        group_num       => 4220,
        short_name      => 'IT_GLOBAL_ENG',
        long_name       => 'TECHNOLOGY_GLOBAL_ENGINEERS',
        parent_name     => 'IT_GLOBAL_MGR'
    );


SA_COMPONENTS.CREATE_GROUP (
        policy_name     => 'general_ols_policy',
        group_num       => 4221,
        short_name      => 'IT_GLOBAL_TEC',
        long_name       => 'TECHNOLOGY_GLOBAL_TECHNICIANS',
        parent_name     => 'IT_GLOBAL_ENG'
    );
```

```
END;

--Authorize Users for the Label Security Policy

BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'hr_1',
            max_level    => 'S',
            min_level    => 'R');

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'hr_1',
            read_comps    => 'MAN,TEC,SAF',
            write_comps   => 'MAN,TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'hr_1',
            read_groups    => 'HR_MGR');
END;

BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'hr_2',
            max_level    => 'S',
            min_level    => 'R'
            );

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'hr_2',
            read_comps    => 'TEC,SAF',
            write_comps   => 'TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'hr_2',
            read_groups    => 'HR_REC',
            row_groups     => 'HR_REC');
END;
```

```
BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'globalit_1',
            max_level    => 'S',
            min_level    => 'R');

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'globalit_1',
            read_comps    => 'MAN,TEC,SAF',
            write_comps   => 'MAN,TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'globalit_1',
            read_groups    => 'IT_GLOBAL_MGR',
            row_groups     => 'IT_GLOBAL_MGR');
END;

BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'globalit_2',
            max_level    => 'C',
            min_level    => 'R');

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'globalit_2',
            read_comps    => 'TEC,SAF',
            write_comps   => 'TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'globalit_2',
            read_groups    => 'IT_GLOBAL_ENG',
            row_groups     => 'IT_GLOBAL_ENG');
END;
```

```
BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'globalit_3',
            max_level    => 'C',
            min_level    => 'R',
            def_level     => 'R',
            row_level     => 'R'  );

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'globalit_3',
            read_comps    => 'TEC,SAF',
            write_comps   => 'TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'globalit_3',
            read_groups    => 'IT_GLOBAL_TEC',
            row_groups     => 'IT_GLOBAL_TEC');
END;

BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'itsupport_2',
            max_level    => 'S',
            min_level    => 'R'
            );

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'itsupport_2',
            read_comps    => 'MAN,TEC,SAF',
            write_comps   => 'MAN,TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'itsupport_2',
            read_groups    => 'IT_SUPPORT_MGR',
            row_groups     => 'IT_SUPPORT_MGR');
END;
```

```
BEGIN
    SA_USER_ADMIN.SET_LEVELS (
            policy_name  => 'general_ols_policy',
            user_name    => 'itsupport_1',
            max_level    => 'C',
            min_level    => 'R',
            def_level     => 'R',
            row_level     => 'R'
            );

     SA_USER_ADMIN.SET_COMPARTMENTS (
            policy_name   => 'general_ols_policy',
            user_name     => 'itsupport_1',
            read_comps    => 'TEC,SAF',
            write_comps   => 'TEC,SAF');

    SA_USER_ADMIN.SET_GROUPS (
            policy_name    => 'general_ols_policy',
            user_name      => 'itsupport_1',
            read_groups    => 'IT_SUPPORT_TEC',
            row_groups     => 'IT_SUPPORT_TEC');
END;
```

Appendix B: OLS Policy Table Setup

The following code represents how the OLS policy created in Appendix A is applied to a table. In this example, the OLS policy is applied to PROJECTS table. The same is done across other tables depending on their use case, but the steps are largely similar.

```
-- Apply OLS policy to PROJECTS table
BEGIN
    SA_POLICY_ADMIN.APPLY_TABLE_POLICY (
            policy_name    => 'general_ols_policy',
            schema_name    => 'company',
            table_name     => 'projects',
            table_options  => 'read_control, write_control, label_default');
END;

-- Enable OLS policy on PROJECTS table
BEGIN
    SA_POLICY_ADMIN.ENABLE_TABLE_POLICY (
            policy_name => 'general_ols_policy',
            schema_name => 'company',
            table_name  => 'projects');
END;
```

For existing data added, the rows can also be updated by the administrator as such to suit the appropriate access control levels. Here is an example from COMPANY.EMPLOYEEBENEFITS

```
UPDATE Company.EmployeeBenefits eb
SET general_ols_col = CHAR_TO_LABEL('general_ols_policy', 'R')
WHERE eb.SecurityLevel = 'Restricted';

UPDATE Company.EmployeeBenefits eb
SET general_ols_col = CHAR_TO_LABEL('general_ols_policy', 'S::HR')
WHERE eb.SecurityLevel = 'Secret';

UPDATE Company.EmployeeBenefits eb
SET general_ols_col = CHAR_TO_LABEL('general_ols_policy', 'C::IT_GLOBAL')
WHERE eb.SecurityLevel = 'Confidential' and DEPARTMENT = 'Global IT';

UPDATE Company.EmployeeBenefits eb
SET general_ols_col = CHAR_TO_LABEL('general_ols_policy', 'C:FIN:HR')
WHERE eb.SecurityLevel = 'Confidential' and DEPARTMENT = 'HR';
```

Appendix C: VPD to facilitate Insert/Update access

VPD on Employees Table

```
CREATE CONTEXT employee_position USING company.context_employee_position_package;
/

CREATE OR REPLACE PACKAGE company.context_employee_position_package AS
  PROCEDURE set_employee_position_context;
END;
/

CREATE OR REPLACE PACKAGE BODY company.context_employee_position_package IS
  PROCEDURE set_employee_position_context IS
    v_username  VARCHAR2(40);
    v_employee_position VARCHAR2(12);
  BEGIN
    v_username := SYS_CONTEXT('USERENV','SESSION_USER');
    BEGIN
      SELECT position
      INTO   v_employee_position
      FROM   COMPANY.EMPLOYEES
      WHERE  UPPER(username) = v_username;

      DBMS_OUTPUT.PUT_LINE('set_employee_position_context position: ' ||
v_employee_position);

      DBMS_SESSION.set_context('employee_position','position', v_employee_position);
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        DBMS_SESSION.set_context('employee_position','position', NULL);
    END;
  END set_employee_position_context;
END context_employee_position_package;
/

GRANT EXECUTE ON company.context_employee_position_package TO PUBLIC;
CREATE PUBLIC SYNONYM context_employee_position_package FOR
company.context_employee_position_package;

GRANT ADMINISTER DATABASE TRIGGER TO COMPANY

CREATE OR REPLACE TRIGGER company.set_employee_position_trigger AFTER LOGON
ON DATABASE
BEGIN
```

```
  company.context_employee_position_package.set_employee_position_context;
END;
/

CREATE OR REPLACE FUNCTION company.employee_view_emplopyee_info(
          p_schema IN VARCHAR2, p_object IN VARCHAR2)
RETURN VARCHAR2 AS
   v_username VARCHAR2(40);
   v_position VARCHAR2(12);
   condition VARCHAR2(200);
BEGIN
   v_username := UPPER(SYS_CONTEXT('USERENV','SESSION_USER'));
   v_position := UPPER(SYS_CONTEXT('employee_position','position'));
   DBMS_OUTPUT.PUT_LINE('v_username: ' || v_username);
   DBMS_OUTPUT.PUT_LINE('v_position: ' || v_position);

  IF (v_username = 'COMPANY' OR v_username = 'SYSTEM' OR v_position = 'MANAGER' OR
v_position = 'RECRUITER') THEN
      DBMS_OUTPUT.PUT_LINE('employee_view_emplopyee_info return condition: ' || '1=1');
      RETURN '1=1';
   ELSE
      condition := 'UPPER(username) = SYS_CONTEXT("USERENV","SESSION_USER")';
      DBMS_OUTPUT.PUT_LINE('employee_view_emplopyee_info return condition: ' ||
condition);
      RETURN condition;
   END IF;

END employee_view_emplopyee_info;

BEGIN
          DBMS_RLS.ADD_POLICY (
          object_schema    => 'COMPANY',
          object_name      => 'EMPLOYEES',
          policy_name      => 'employee_view_emplopyee_info_policy',
          function_schema  => 'COMPANY',
          policy_function  => 'employee_view_emplopyee_info',
       update_check     => true
           );
END;
```

Appendix D: Database Table Creation Code

```
CREATE TABLE EmployeeTrainingRecords (
   TrainingRecordID    NUMBER PRIMARY KEY,
   EmployeeID          NUMBER REFERENCES Employees(EmployeeID),
   TrainingCerts       VARCHAR2(255),
   SecurityLevel       VARCHAR2(50),   -- e.g., 'Restricted', 'Confidential', ‘Secret’
   TrainingType        VARCHAR2(100),  -- Compartment (e.g., 'Technical', 'Managerial',
'Safety')
   Position            VARCHAR2(100)   -- Group (e.g., 'Manager', 'Engineer', 'Technician')
);
```

```
CREATE TABLE ProjectDocumentation (
   DocID               NUMBER PRIMARY KEY,
   ProjectName         VARCHAR2(255),
   DocumentType        VARCHAR2(100),  -- e.g., 'Technical', 'Financial', 'Strategy'
   Content             CLOB,
   SecurityLevel       VARCHAR2(50),   -- e.g., 'Restricted', 'Confidential', ‘Secret’
   Department          VARCHAR2(100),  -- Group; e.g., ‘Global IT’, 'HR', ‘IT Support’
   DataType            VARCHAR2(100)   -- Compartment; can be 'Report', 'Blueprint', 'Budget'
);
```

```
CREATE TABLE Employees (
   EmployeeID     NUMBER PRIMARY KEY,
   UserName       VARCHAR2(100),
   FirstName      VARCHAR2(100),
   LastName       VARCHAR2(100),
   Email          VARCHAR2(150) UNIQUE,
   DateOfJoining  DATE,
   SecurityLevel       VARCHAR2(50),   -- e.g., 'Restricted', 'Confidential', ‘Secret’
   Position       VARCHAR2(100),  -- Group; e.g., 'Manager', 'Engineer'
   Department     VARCHAR2(100)   -- Group; ‘HR’, ‘Global IT, IT Support
);
```

```
CREATE TABLE Departments (
   DepartmentID   NUMBER PRIMARY KEY,
   DepartmentName VARCHAR2(100)
);

– Departments: HR, Global IT (Parent Group), IT Support (Child Group)
```

```
CREATE TABLE Projects (
   ProjectID      NUMBER PRIMARY KEY,
   ProjectName    VARCHAR2(255),
   StartDate      DATE,
   EndDate        DATE,
   Department     VARCHAR2(100)   -- Group; ‘HR’, ‘Global IT, IT Support
);
```

```
CREATE TABLE EmployeeBenefits (
   BenefitID      NUMBER PRIMARY KEY,
   BenefitType    VARCHAR2(100),  -- e.g., 'Health Insurance', 'Stock Option'
   Details        VARCHAR2(500),
   SecurityLevel  VARCHAR2(50),   -- e.g., 'Restricted', 'Confidential', ‘Secret’
   Department     VARCHAR2(100),  -- Group; HR, Global IT, IT Support
   BenefitForm    VARCHAR2(100)   -- Compartment; can be 'Medical', 'Financial',
'Retirement'
);
```

```
CREATE TABLE MeetingNotes (
   NoteID         NUMBER PRIMARY KEY,
   MeetingDate    DATE,
   Topic          VARCHAR2(255),
   Content        CLOB,
   SecurityLevel  VARCHAR2(50),   -- e.g., 'Restricted', 'Confidential', ‘Secret’
   Department     VARCHAR2(100),  -- Group; HR, Global IT, IT Support
   MeetingType    VARCHAR2(100)   -- Compartment; e.g., 'Strategy', 'Budget', 'Review'
);
```