Lab: Oracle Unified Auditing

Objective:

To enable unified auditing, create unified audit policies, and maintain the audit trail. This lab will explain each step and provide validation checks. All actions will be performed in SQL Developer unless otherwise specified.

Prerequisites:

- Oracle Database is installed and running.
- Unified auditing is not yet enabled.

Steps:

1. Enable Unified Auditing

Explanation: Unified auditing consolidates all audit records into a single audit trail, making it easier to manage and review audit information.

1. **Check the Current Audit Configuration:**
    - **SQL*Plus (Terminal):**
        - Connect to the database as SYSDBA and check the current audit configuration.

```
sqlplus / as sysdba
SHOW PARAMETER audit_trail;
```

2. **Enable Unified Auditing:**
    - **Terminal:**
        - Unified auditing must be enabled at the time of database creation or by relinking the Oracle binaries. To enable it by relinking:

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk uniaud_on
```

3. **Restart the Database:**
    - **SQL*Plus (Terminal):**
        - Restart the database to apply changes.

```
SHUTDOWN IMMEDIATE;
STARTUP;
```

4. **Verify Unified Auditing is Enabled:**
    - **SQL Developer:**
        - Connect to the CDBLAB as SYSDBA and run the following query:

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
```

2. Create Unified Audit Policies

Explanation: Unified audit policies allow you to specify which actions to audit and under what conditions. This granularity helps in monitoring specific database activities.

1. **Create a Table for Auditing:**
   - **SQL Developer:**
     - Connect to PDBLAB1 as SYSDBA and create a table named `EMPLOYEES`.

```
CREATE TABLE EMPLOYEES (
  EMP_ID NUMBER PRIMARY KEY,
  NAME VARCHAR2(50),
  DEPARTMENT VARCHAR2(50),
  SALARY NUMBER
);
```

2. **Insert Data into the Table:**
   - **SQL Developer:**
     - Insert sample data into the `EMPLOYEES` table.

```
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES
(1, 'John Doe', 'IT', 70000);
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES
(2, 'Jane Smith', 'HR', 60000);
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES
(3, 'Alice Johnson', 'Finance', 75000);
COMMIT;
```

3. **Create an Audit Policy for DML Operations:**
   - **SQL Developer:**
     - Create a policy named `DML_AUDIT_POLICY` to audit DML operations on the `EMPLOYEES` table.

```
CREATE AUDIT POLICY DML_AUDIT_POLICY
ACTIONS INSERT, UPDATE, DELETE ON EMPLOYEES;
```

4. **Create an Audit Policy for User Logins:**
   - **SQL Developer:**
     - Create a policy named `LOGIN_AUDIT_POLICY` to audit user login attempts.

```
CREATE AUDIT POLICY LOGIN_AUDIT_POLICY
ACTIONS LOGON, LOGOFF;
```

5. **Enable the Audit Policies:**
   - **SQL Developer:**
     - Enable the created audit policies.

```
AUDIT POLICY DML_AUDIT_POLICY;
AUDIT POLICY LOGIN_AUDIT_POLICY;
```

6. **Verify the Audit Policies:**

    - **SQL Developer:**
        - Check the enabled audit policies.

        ```
        SELECT * FROM AUDIT_UNIFIED_ENABLED_POLICIES;
        ```

3. Maintain the Audit Trail

Explanation: Maintaining the audit trail involves reviewing audit records and managing the storage of audit data to ensure it does not impact database performance.

1. **View Audit Records:**

    - **SQL Developer:**
        - Query the unified audit trail to view audit records.

        ```
        SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME, OBJECT_SCHEMA,
        OBJECT_NAME
        FROM UNIFIED_AUDIT_TRAIL;
        ```

2. **Purge Old Audit Records:**

    - **SQL Developer:**
        - Purge old audit records to maintain performance and storage.

        ```
        BEGIN
        DBMS_AUDIT_MGMT.INIT_CLEANUP(
          AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
          USE_LAST_ARCH_TIMESTAMP => TRUE
        );
        END;
        /
        ```

3. **Set a Retention Period for Audit Records:**

    - **SQL Developer:**
        - Set a retention period to automatically purge audit records older than a specific duration.

        ```
        BEGIN
        DBMS_AUDIT_MGMT.SET_PURGE_JOB(
          AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
          AUDIT_TRAIL_PURGE_INTERVAL => 24
        );
        END;
        /
        ```

Validations:

1. **Check Current Audit Configuration:**

- **SQL*Plus (Terminal):**

```
SHOW PARAMETER audit_trail;
```

2. **Verify Unified Auditing Enabled:**
   - **SQL Developer:**

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
```

3. **Verify Enabled Audit Policies:**
   - **SQL Developer:**

```
SELECT * FROM AUDIT_UNIFIED_ENABLED_POLICIES;
```

4. **View Audit Records:**
   - **SQL Developer:**

```
SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME, OBJECT_SCHEMA,
OBJECT_NAME
FROM UNIFIED_AUDIT_TRAIL;
```

Summary:

By following these steps, you will enable unified auditing, create unified audit policies, and maintain the audit trail. This lab enhances your ability to monitor and manage database activities, ensuring compliance with security and regulatory requirements. Each step is explained with its purpose, and validations are provided to ensure correct implementation.

Addendum: Testing and Demonstrating Unified Auditing

Objective:

To test and demonstrate the effectiveness of the unified auditing setup by performing actions that are audited and verifying the audit trail for those actions.

Steps:

1. Perform Audited Actions

Explanation: To validate the audit policies, we need to perform actions that will trigger the auditing mechanisms we set up.

1. **Insert Data into the EMPLOYEES Table:**
   - **SQL Developer:**
     - Connect to PDBLAB1 as `dev_user`.
     - Insert new records into the `EMPLOYEES` table.

```
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES
(4, 'Bob Brown', 'Marketing', 65000);
INSERT INTO EMPLOYEES (EMP_ID, NAME, DEPARTMENT, SALARY) VALUES
(5, 'Carol White', 'Sales', 62000);
COMMIT;
```

2. **Update Data in the EMPLOYEES Table:**
    - **SQL Developer:**
        - Update records in the `EMPLOYEES` table.

```
UPDATE EMPLOYEES SET SALARY = 67000 WHERE EMP_ID = 1;
COMMIT;
```

3. **Delete Data from the EMPLOYEES Table:**
    - **SQL Developer:**
        - Delete a record from the `EMPLOYEES` table.

```
DELETE FROM EMPLOYEES WHERE EMP_ID = 2;
COMMIT;
```

4. **Log In and Log Off:**
    - **SQL Developer:**
        - Log in and log off as different users to generate login and logoff events.

2. Verify Audit Records

Explanation: After performing the audited actions, we need to verify that these actions have been recorded in the audit trail.

1. **Query the Unified Audit Trail:**
    - **SQL Developer:**
        - Connect to PDBLAB1 as SYSDBA and query the unified audit trail to view the audit records.

```
SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME, OBJECT_SCHEMA,
OBJECT_NAME
FROM UNIFIED_AUDIT_TRAIL
WHERE OBJECT_NAME = 'EMPLOYEES';
```

2. **Check for Login and Logoff Events:**
    - **SQL Developer:**
        - Query the unified audit trail to view login and logoff events.

```
SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME
FROM UNIFIED_AUDIT_TRAIL
WHERE ACTION_NAME IN ('LOGON', 'LOGOFF');
```

3. Validate the Audit Policies

Explanation: Ensure that the audit policies are capturing the intended actions and providing the necessary information.

1. **Validate DML Audit Policy:**
    - **SQL Developer:**
        - Verify that DML operations (INSERT, UPDATE, DELETE) on the `EMPLOYEES` table are being audited.

```
SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME, OBJECT_SCHEMA,
OBJECT_NAME, SQL_TEXT
FROM UNIFIED_AUDIT_TRAIL
WHERE OBJECT_NAME = 'EMPLOYEES';
```

2. **Validate Login Audit Policy:**
    - **SQL Developer:**
        - Verify that login and logoff events are being audited.

```
SELECT EVENT_TIMESTAMP, DBUSERNAME, ACTION_NAME
FROM UNIFIED_AUDIT_TRAIL
WHERE ACTION_NAME IN ('LOGON', 'LOGOFF');
```

4. Demonstrate Audit Trail Maintenance

Explanation: Show how to maintain the audit trail by purging old records and setting retention policies.

1. **Purge Old Audit Records:**
    - **SQL Developer:**
        - Purge audit records older than a specified timestamp.

```
BEGIN
DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(
  AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
  USE_LAST_ARCH_TIMESTAMP => TRUE
);
END;
/
```

2. **Set Audit Record Retention Period:**
    - **SQL Developer:**
        - Set a retention period for the audit records.

```
BEGIN
DBMS_AUDIT_MGMT.SET_RETENTION(
  AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
  RETENTION_VALUE => 90
);
```

```
END;
/
```

Summary:

By following this addendum, you will perform actions that trigger the auditing mechanisms, verify that the actions are recorded in the audit trail, and demonstrate how to maintain the audit trail. This ensures that the unified auditing setup is functioning correctly and providing the necessary monitoring and security for database activities.

Each step includes the necessary commands and explanations to ensure clear understanding and proper execution.